AI for Detecting Hidden Cyberthreats in Encrypted Data

Final Year Project Report

Obaid Sheikh Ahmed	**(BAI-21F-012)**
Arsalan	**(BAI-21F-003)**
Mohammad Talha Panhwar	**(BAI-21F-008)**

SESSION 2021-2025

Supervisor

Syed Azeem Inam

Project report in partial fulfillment of the requirements for the award of Bachelor of Science (Artificial Intelligence) degree.

IN

DEPARTMENT OF ARTIFICIAL INTELLIGENCE

SINDH MADARESSATUL ISLAM UNIVERSITY KARACHI, PAKISTAN

(September 2025)

Declaration

We hereby declare that this project report is based on our original work except for citation and quotation which have been duly acknowledged. We also declare that it has not been previously and concurrently submitted for any other degree or award at SINDH MADARESSATUL ISLAM UNIVERSITY or other institute.

1) Name: Obaid Sheikh Ahmed (BAI-21F-012)

Signatures: _________________

2) Name: Arsalan (BAI-21F-003)

Signatures: _________________

3) Name: Muhammad Talha Panhwar (BAI-21F-008)

Signatures: ________________

AI for Detecting Hidden Cyberthreats in Encrypted Data

Approval For Submission

THE DEPARTMENT OF ARTIFICIAL INTELLIGENCE, SINDH MADRESSATUL ISLAM UNIVERSITY, ACCEPTS THIS THESIS SUBMITTED BY Obaid Sheikh Ahmed, Arsalan and Muhammad Talha Pnahwar IN ITS PRESENT FORM AND IT IS SATISFYING THE DISSERTATION REQUIREMENTS FOR THE AWARD OF BACHELOR DEGREE IN ARTIFICIAL INTELLIGENCE.

SUPERVISOR:

SYED AZEEM INAM

Designation ____

HOD OF DEPARTMENT:

DR. MANSOOR KHURO ______________________________

DEAN/HOD OF DEPARTMENT:

DR. AFTAB AHMED SHAIKH _________________________

DATED: _______________

DEPARTMENT OF ARTIFICIAL INTELLIGENCE

SINDH MADRESSATUL ISLAM UNIVERSITY,

KARACHI, PAKISTAN

Abstract

The use of encrypted network traffic is rapidly increasing day by day. This creates challenges for traditional Intrusion Detection Systems (IDS) because they faces lot of difficulties in facing the analyzation of encrypted flows. This study has addressed this issue by developing an AI based system which can detect hidden cyber threats from the traffic flows. The importance of this research is to ensure data security and to ensure the user privacy and provide scalable solutions for modern digital environments like Iot, cloud and enterprise network. Two datasets used in this study, CICID 2017 and UNSW-NB15. These datasets contain multiple attacks and flow level features. Traditional Machine Learning and deep learning models applied on both datasets. A hybrid model is proposed which use autoencoder to learn the traffic patterns of the data and XGBoost classifier, which performed best on CICID 2017 dataset, is used to predict attacks. Result shows that XGBoost performed best on CICID 2017 dataset achieving the accuracy of 98.62% and Random Forest classifier performed best on UNSW-NB15 dataset achieving the accuracy of 99.23%. Hybrid model gave results better than the traditional models applied on CICID2017 dataset. For future implementation this AI model is deployed on cloud hosted platform, which provide detection from the rest API and real time monitoring from website.

Acknowledgement

In the name of Allah, the most Gracious and the Most Merciful.

Peace and blessing of Allah be upon Prophet Muhammad ﷺ

Firstly, we would like to thanks Allah SWT for blessing us with the hope of this new idea and then We would like to express our deepest gratitude and appreciation to **Sir Syed Azeem Inam**, our dedicated and supportive course supervisor, for his invaluable guidance and unwavering encouragement throughout the completion of our Final Year Project (FYP). His expertise, insightful feedback, and commitment to our academic growth have been instrumental in shaping the outcome of this project.

_______________ meticulous attention to detail and his ability to provide constructive criticism helped refine our research methodology, strengthen our analytical approach, and enhance the overall quality of this documentation. His patience, accessibility, and willingness to invest time and effort in addressing our queries and concerns played a significant role in our collective personal and professional development.

Furthermore, we are deeply grateful to our families and friends for their unwavering support, understanding, and encouragement throughout this challenging endeavor. Their belief in our abilities and their constant motivation have been a constant source of inspiration.

This project would not have been possible without the collective efforts and support of these individuals and many others who have contributed to our academic and personal growth. We are truly honored and humbled by their presence in our journey.

Lastly, we would like to express our heartfelt appreciation to all the participants who willingly participated in the data collection process, enabling us to gather valuable insights and conduct a comprehensive analysis.

Dedication

Alhamdulillah, it’s all because of Allah (SWT) that I have been able to accomplish this Final Year Project with patience, resilience, and, most importantly, with Allah’s guidance, good health and knowledge. I can’t imagine how difficult this would have been without Allah’s direction each step of the way.

The real drive behind this effort and focus comes from my parents. I want to thank them and dedicate this project to them as it is their unfailing love, prayers, encouragement, and sacrifices that have helped to bring up my personal and academic growth. I can see that everything I have been able to accomplish at this level is only because of their constant support and faith in me.

My appreciation also goes to my supervisor and teachers from my obtaining the knowledge and being guided to the right path. Their constructive criticism, suggestions, and all the encouragement along the way have greatly assisted in learning during this project.

My friends and well wishers, with their enthusiasm around me make it possible to focus and be inspired to rise above the most challenging situations. I thank them all and will dedicate this project.

Finally, it is with great hope and respect that I direct this work to all them who have great passion in acquisition of knowledge like my work decent contribution in the Artificial Intelligence field.

Table Of Contents

Chapter 1

Introduction

1.1 Historical background:

Cyber Security becomes the important part of life form the day internet was introduced to people. Protecting someone's private and sensitive data is important as digital systems taking place over global operations. E.g: Protecting online data of every individual, financial transaction, national defence networks and healthcare records. Over the time the security of stuff available on internet is enhancing day by day which includes firewalls along with software of antivirus and Intrusion Detection Systems (IDS). The working of all these techniques is to block the known threats effectively. Whereas technology is increasing so as the cyber attacks increasing its intensity on the sensitive data. Due to this, organizations have no choice but to work more and more on encryption techniques. Traditional security tools can't scan encrypted traffic properly, so attackers started using these secure channels themselves hiding malicious payloads inside wrapped in layers of unreadable code ironically turning protection into vulnerability. Situations like these makes the cyber security team in more stress, leading them to work more on protection of data of any organization. As organizations faced many difficulties in protecting data why not make a smart security tool? In the field of Information Technology, Artificial Intelligence is a boost to any work which take human efforts and time. Security purpose works, if done with Artificial Intelligence, will makes work easy. Machine learning ML and Deep learning DL are leading the charge here with AI stuff. In sense of cyber security, ML and DL train the model of Ai application or software which can find out the malicious activities based on there training samples parameters and entries. The model after training see the real time patterns and detect the suspicious activity. For example: Timing, packet sizes, those statistical hiccups that don't add up normally. All while keeping the actual data locked tight. The technology is not perfect obviously. But being able to flag potential threats just by watching how data moves around changes things big time. Especially when you consider how much encrypted traffic we're dealing with daily now.

1.2 Keywords Explanation

1.2.1 Artificial Intelligence

The capability of a machine to do tasks same as human do, is . Some tasks requires human intelligence which can be done by machines and this is because of Artificial Intelligence. Artificial intelligence (AI) models are used in cybersecurity to identify patterns of existing threats, identify anomalies, and even forecast undiscovered or zero-day threats. AI has become a potential weapon for data protection against the cyber threats of modern days.

1.2.2 Cyber Threats

A malicious activity that threatens to damage or steal data from a system, is known as Cyber Treats, which consists Distributed Denial of Service (DdoS) attacks, phishing attacks, advanced persistent threats and ransomware. These attacks are dangerous for anyone's systems and these threats becoming more stronger day by day making challenging task to security providers.

1.2.3 Encrypted Data

The human understandable data or any information that has been transformed to unreadable form to protect it from unauthorized access, is known as Encrypted data. Data integrity and confidentiality are therefore guaranteed. Some popular encryption standards are SSL/TLS, RSA, and AES. Although encryption shields information from unwanted access, it also prevents conventional security tools from scanning payloads for dangers.

1.2.4 Hidden Threats

Some of the threats that are purposefully hidden to avoid detection and appears suddenly, are known as Hidden Threats. Backdoors encoded in encrypted communication, malware command-and-control (C2) signals, and data exfiltration efforts are examples of hidden risks that might affect encrypted data. It is necessary to examine traffic flow, metadata, and behavioral irregularities in order to identify such threats without decrypting the content.

1.3 Objectives of the Present Study

The objectives of this study are, building a machine learning model using encrypted traffic datasets. They will test how accurate it is at catching sneaky threats without disturbing legitimate traffic. Then there's checking which one actually works best in real setups either it is from deep learning or from traditional machine learning approaches. Second objective of this study is to combine different techniques for better results. Like using an autoencoder for spotting weird patterns paired with XGBoost for classifying threats. Whole point is making something that's both efficient and reliable compared to existing tools out there and judging the performance of different models by standard stuff like accuracy, precision rates and recall percentages across different attack scenarios. Benchmarking against current methods comes next. After comparing the best performing model on the datasets, now comes a part to compare Hybrid model with the best performing traditional model to see if the hybrid model actually improves detection rates while keeping false alarms low. Third objective of this study is to deploy our model on virtual machine and show the results of the best performing model in website while using random data use for cyber threats, a system that works in real-world networks without slowing things down or requiring massive computing power compared to older detection systems.

1.4 Novelty of Study

Most intrusion detection setups right now depend on checking content directly but know that approach falls apart when dealing with encrypted data. This study aims to make an AI based system that detects cyber threats within the Encrypted Data. Traditional detection of threats usually decrypt the encrypted data to understand where the problem is but AI system made in this study will keeps user privacy safe while delivering quicker solutions that can handle real-world scale. The system in this study uses deep learning models for example CNNs and Transformers. These analyze patterns within encrypted traffic flows which are not common in regular security tools yet. This study is focused purely on traffic patterns instead of peeking at actual data payloads. The train models of Ai based system will detect the malicious activities from the new traffic from any server or a website. The benefit is that suspicious activities can be detect within the encrypted data without compromising privacy, which has been a sticking point for years now. Validation covered multiple attack types and network conditions showing consistent performance metrics. Data sources included real-world traffic samples to test the model's flexibility. Two datasets were selected in this study to train and test which model works well so the best performing model will be the key solution of AI system. Scalability improvements come naturally from processing patterns instead of content depth which could be huge for cloud environments handling insane traffic volumes daily. The usage of Deep Learning is because there were complex attack signatures that rule-based systems might miss especially in encrypted formats where payloads look random. Combine feature engineering tricks with the model and model training to detect new patterns and stay relevant as it faces new patterns over time. The benefit of this approach is that it will not replace the existing tools, but it will act as another layer of security where the traditional tools might detect any new pattern incorrectly. Mostly detection tools depend on checking the decrypted content and then find suspicious activity in the data, but that approach falls apart when dealing with encrypted data. This study aims to make an AI based System which detect hidden patterns from the data without decrypting it. It keeps users safe and delivers quick solutions that can be done on real world scenario.

1.5 Problem Statement

The usage of internet is increasing day by day and browsing websites increase the traffic on any website, Encrypted traffic keeps blowing up these days. Here’s the thing though, most tools right now can’t even peek into encrypted stuff without decrypting first. Which is a problem for two big reasons. It slows everything or else it needs a high computational system to perform it quickly because if the system on which an application or software is running, late work might be the positive point for hackers to perform their work quickly and run before the system responds. It straight up breaks privacy laws like GDPR or HIPAA. Hackers totally know this weakness too. They hide their attack in encrypted tunnels more than ever now. Current technology is missing something crucial. To solve this gap, it is important to make a system that can spot bad patterns without decrypting data. This study shows that AI is thrown at the problem but keeps things private. The system detects suspicious activity if any malicious things happens while leaving the actual data sealed up tight. People always want both security and privacy. This might pull off that balancing act for once.

Chapter 2

Literature Review

Network intrusion detection used to rely on signature based systems back in the day. Now it’s shifted more toward anomaly based frameworks that use stuff of machine learning and deep learning. Early methods depended heavily on predefined signatures. Those worked well for spotting threats they already knew about but couldn’t handle new attack vectors at all really. All those limitations ended up driving a move toward statistical anomaly detection instead of relying on old methods. Supervised learning approaches started gaining ground around that time, people noticed how useful Supervised learning approaches were becoming in machine learning algorithms. It really took off here stuff like decision trees SVM models k nearest neighbors all that became pretty common, main reason was their ability to take labeled examples and apply them more broadly, older systems couldn’t do that as effectively, so this made a big difference over time. People realized these methods could actually adapt better when faced with new types of attacks which was kind of a game changer even if it wasn’t perfect right away there were still challenges obviously but compared to signature based approaches this felt like progress finally happening after years of stagnation in the field. Deep learning taking things even further down that path today’s systems are way more flexible because of all that groundwork laid back then by those earlier machine learning techniques honestly without those steps forward we would probably still be stuck dealing with way more false positives and missed threats across networks globally. Some techniques like Random Forest and gradient-boosted models (like XGBoost and LightGBM) are machine learning-based approaches that have continuously produced great performance in intrusion detection applications in recent studies. Random Forest classifier and Extra Trees classifiers have showed great accuracy in a recent comparative analysis across the UNSW-NB15, CIC-IDS2017, and CIC-IDS2018 datasets, attaining 99.59% and 99.95% on UNSW-NB15, respectively, and even 99.99% on CIC-IDS2017 under ten-fold cross-validation [1]. Deep Learning which use in data like time-series and location data, now a days used in traffic data. A light model CNN_BiLSTM is used in a recent study, achieving the accuracy of 97.28% in detecting intrusion from normal traffic [2]. Similarly, another study showed that, when applied an attention-augmented LSTM architecture (AT-LSTM) on UNSW-NB15, it perform better than other models that was used in the study, achieving the accuracy of 92.2%. CNN-BiLSTM applied to UNSW-NB15 resulting in the result with the accuracy of 77% and Rule-Based system achieve accuracy of 57% [3]. Furthermore, another study uses the data of UNSW-NB15 and Network Security Laboratory Knowledge Discovery in Databases (NSL-KDD) and applied a Neural Network DNN model to get a 80-90% of accuracy. This study shows that if DNN is not applied carefully then there will be an risk of overfitting which results in weak performance [4].

In study of 2021, it is shown that when Network traffic data is converted in binary form using finite state machine technique, it improved the performance of LSTM and GRU models by 13-18% and F1 score of the dataset also improved [5]. There is still some gap in researches that is mostly studies show the testing on one dataset, due to which general use of traffic pattern isn’t clear. Pre-processing and Feature selection are always different in every study that makes it difficult to compare results. For example, a study in 2022 shows that changing features from 42 to 23 resulting in the improvement of MLP model by achieving accuracy of 84.2% from 82.3% but this method does not apply on other studies [6]. As mentioned above that mostly studies test there work on one dataset because each model perform differently on different datasets. In this study its also showed that the model trained on CICIDS2017 dataset, it results in good performance of model but the model applied to other datasets, it results bad. That's why cross-dataset testing is important [7]. A recent study of 2024 shows that some models give high accuracy on training dataset but when those models test on other dataset, there accuracies fall down, proving that real-world reliability is often overestimated [8].

2.1 Cross Data Generalization

Many intrusion detection systems based on machine learning (ML-IDS) perform well when trained on the same dataset and tested on the same dataset, such as CIC-IDS-2017. Sometime their performance drops when they are applied to see new patterns(unseen dataset) like UNSW-NB15 or ISCX-12. For example, de Carvalho Bertoli et al. (2024) conducted a cross-dataset evaluation using multiple NetFlow-based datasets and observed a decline of up to 30% in AUROC when models were tested on datasets different from those they were trained on. This shows overfitting creates issue and a lack of generalizability in current approaches[8].

2.1.1 How this study fills the gap

This study explicitly trains and evaluates models separately on the datasets of CICIDS2017 and UNSW-NB15 without merging them. By keeping the datasets independent, the study is able to demonstrate the generalizability of the hybrid model across different network environments. This method solve the dataset overfitting problem and poor generalization issues highlighted in previous research.

2.2 Privacy-Respecting Detection in Encrypted Traffic

Normally intrusion detection systems depends on deep packet inspection (DPI), which do not produce any significant or desired effect in encrypted environments and raises privacy concerns. Many solutions were taken to decrypt traffic for analysis, this violates the privacy laws and increases computational power. A survey was published in 2021, which showed the existing privacy-preserving techniques and concluded that there is a significant gap between academic proposals and practical, scalable solutions for detecting threats in encrypted traffic[9].

2.2.1 How this study fills the gap

This study uses only flow-based features such as duration, packet size and inter-arrival times to detect suspicious activities in encrypted traffic without checking payload entirely, this is how the system maintains user privacy and achieve high accuracy. The model is hybrid architecture, combines from a deep learning-based autoencoder for feature extraction with an XGBoost classifier for final prediction, this provides an efficient and privacy friendly solution for real time work.

2.3 Vast amount of work in same way in same study

Most of the studies works on one dataset with single or multiple results, due to which it is a bit difficult to search which model is best performing and the comparison between them is difficult because each study have done there all works in there own way.

2.3.1 How this study fill the gap

This study focus on the training and testing of multiple traditional models on two datasets (CICIDS2017 and UNSW-NB15) and comparing the best performing model among both datasets so that it will be easy to understands and search for the best performing model for further working if anyone wants to take over.

Chapter 3

Datasets

3.1 Dataset Description

The performance of the proposed artificial intelligence-based detection model for identifying hidden cyber threats in encrypted data was evaluated using two publicly available and highly reputable datasets (CICIDS 2017 and UNSW-NB15). The reason to select both datasets were because of there detailed information in it, varieties of attack types through which an AI model can have more values to learn resulting in better results. These datasets are good for testing and training of machine learning and deep learning models, particularly in contexts where payload inspection is either not feasible or violates data privacy principles.

3.1.1 CICID2017 Dataset

The Canadian Institute for Cybersecurity developed a dataset in 2017 which is known as CICID 2017. This dataset is made of real world network traffic values which include Bengin and Attack. This dataset contains variety of threats vectors such as Brute Force, DdoS, Botnet and attacks based on Web also known as web based attacks. Each flow record contains a detailed set of flow level features like flow duration, packet counts, mean packet length, and inter arrival times. These features are useful for analyzing the behavior of network without relying on packet payloads so that the dataset help model to learn in a way that the model don't need to decrypt the system to check the attacks. This dataset is used in this study to train machine learning and deep learning models and test them and do compare which model is the best performing among all models on this dataset.

3.1.2 UNSW-NB15 Dataset

UNSW-NB15 dataset is the second dataset which is used in this study. This dataset is developed by Australian Centre for Cyber Security. This dataset was created in 2015. The reason to use this dataset is because it has variety of attacks that are used in training and testing models for AI, where need of decryption don't take place and this dataset has a detailed coverage of modern cyber threats generated using the IXIA Perfect Storm tool. With 49 features per flow, such as duration, packet size, protocol type, and TCP flags, the dataset includes more than two million entries. It covers several different kinds of attacks, including worms, reconnaissance, fuzzers, and exploits. The dataset is appropriate for intrusion detection even in encrypted contexts because of these flow-level properties, which make it possible to learn traffic behavior patterns effectively without depending on payload inspection. The suggested deep learning model performance in SDN-based IoT network scenarios was assessed in this study using UNSW-NB15.

3.1.3 Necessary Details of recent study that utilize these datasets for their studies

A recent study presented deep learning based intrusion detection system, the reason of that study was to present a securing environment for Iot and SDN, using CICIDS 2017 dataset. A Long Short Term Memory (LSTM) model was developed by researchers that efficiently analyzed flow based features without decryption, making suitable for encrypted traffic scenarios. Their system was strictly evaluated against various modern cyberattacks such as DdoS, Brute Force, and Botnet intrusions and achieve great accuracy of 99.9% and a minimal false positive rate. This study shows that temporal deep learning models like LSTM works strong on real time network. Additionally it shows that CICIDS2017 dataset is useful to train IDS training[10].

A a recent study conducted an evaluation of several RNN-based models—including LSTM, bidirectional LSTM (BiLSTM), GRU, and bidirectional GRU—for binary intrusion detection using the UNSW-NB15 dataset, addressing severe class imbalance by applying SMOTE oversampling. They trained and tested each model for classifying network flows as normal or attack and measured performance by observing accuracy, precision, recall, F1-score, and AUC. The outcomes of that study highlighted that the BiLSTM model achieve the best performance, reaching approximately **95.01 %** accuracy and an AUC of **0.9953** after SMOTE application, thus demonstrating its superiority in detecting intrusions within IoT network traffic environments[11].

To better understand the distribution and behavior of the data used for model training and evaluation, flow-level features such as flow duration, packet size mean, total forward and backward packets, and flow inter-arrival times were analyzed. Boxplots and correlation heatmaps were used to illustrate these characteristics, emphasizing the notable statistical distinctions between malicious and normal flows. In the CICIDS 2017 dataset, for example, anomalous increases in flow byte rates and inter-arrival periods were powerful predictors of botnet and DdoS attacks. Similarly, backdoor and reconnaissance attacks were linked to irregularities in protocol distributions and flow lengths in the UNSW-NB15 dataset. Such feature behaviors served as input signals to the machine learning models developed in this study. In addition to offering a variety of attack scenarios, both databases enable the creation of models that protect privacy. Because the goal of this research is to analyze encrypted traffic without decrypting it, the use of statistical flow-based features is in line with ethical and regulatory data-handling guidelines like GDPR and HIPAA. Furthermore, the CICIDS 2017 and UNSW-NB15 datasets were individually preprocessed by merging their respective internal files from their original zip archives to form two complete, unified datasets. Without combining the two datasets, this method guarantees that the entire spectrum of attack types and typical traffic patterns from each dataset are successfully recorded, enabling the model to learn from more complex and extensive network scenarios. A recent study published in Scientific Reports which applied deep learning and XGBoost techniques on CICIDS-2017 and UNSW-NB15 datasets using flow-based features for intrusion detection. The model is useful for encrypted contexts like SDN and IoT because it reached over 99% accuracy. By doing away with the need for payload data, this method preserves privacy while preserving excellent detection performance. The model is useful for encrypted contexts like SDN and IoT because it reached over 99% accuracy. By doing away with the need for payload data, this method preserves privacy while preserving excellent detection performance[12].

Using both UNSW-NB15 and CICIDS 2017, this study guarantees a solid basis for assessing the model’s performance in practical settings. These datasets are excellent options for creating and evaluating AI systems meant to identify covert cyberthreats because of their flow-based information, current attacks, and compatibility with encrypted settings.

Chapter 4

Methodology

The methodology of this research focuses on preparing a robust and structured dataset for training machine learning models capable of detecting hidden cyber threats in encrypted network traffic. This process involved several critical stages, beginning with the collection of 2 datasets namely CICIDS2017 and UNSW-NB15, merging them, followed by intensive preprocessing, cleaning, and feature selection. To initiate the process, the CICIDS2017 and UNSW-NB15 datasets were selected due to there richness and diversity in representing real-world network traffic and cyberattacks.

4.1 Dataset Merging

4.1.1 Merging CICID2017 Dataset

The original structure of the CICIDS2017 dataset was distributed across several CSV files, each corresponding to different days and types of attacks such as DdoS, Port Scan, Web Attacks, and Infiltration attempts. These files included “Friday-Working Hours-Afternoon-Ddos.pcap_ISCX.csv,” “Friday-Working Hours-Afternoon-PortScan.pcap_ISCX.csv,” “Friday-WorkingHours-Morning.pcap_ISCX.csv,” “Monday-Working Hours.pcap_ISCX.csv,” “Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv,” “Thursday-Working Hours-Morning-WebAttacks.pcap_ISCX.csv,” “Tuesday-Working Hours.pcap_ISCX.csv,” and “Wednesday-working Hours.pcap_ISCX.csv.” Each of these files was individually read using Python’s pandas library, and all were merged together to form a unified dataset. This merged dataset contained approximately 3,119,345 records with 85 columns, combining traffic from various attack vectors and benign communications into a single structured data file.

4.1.2 Merging UNSW-NB15 Dataset

Similarly the dataset of UNSW-NB15 is also divided into four separate parts: UNSW-NB15_1, UNSW-NB15_2, UNSW-NB15_3, and UNSW-NB15_4. To create a comprehensive and balanced dataset, all four parts were merged into a single unified file. Combining these segments ensured a diverse distribution of both normal and malicious traffic patterns, enhancing the robustness and generalizability of the machine learning models trained on it. This merged dataset was then used for further preprocessing and analysis. Each dataset contains 700001 rows and 49 columns. The names of columns in each dataset is same with the same order with the same values in it. Combining all datasets together to make a merge dataset of UNSW-NB15.

4.2 Data Analysis

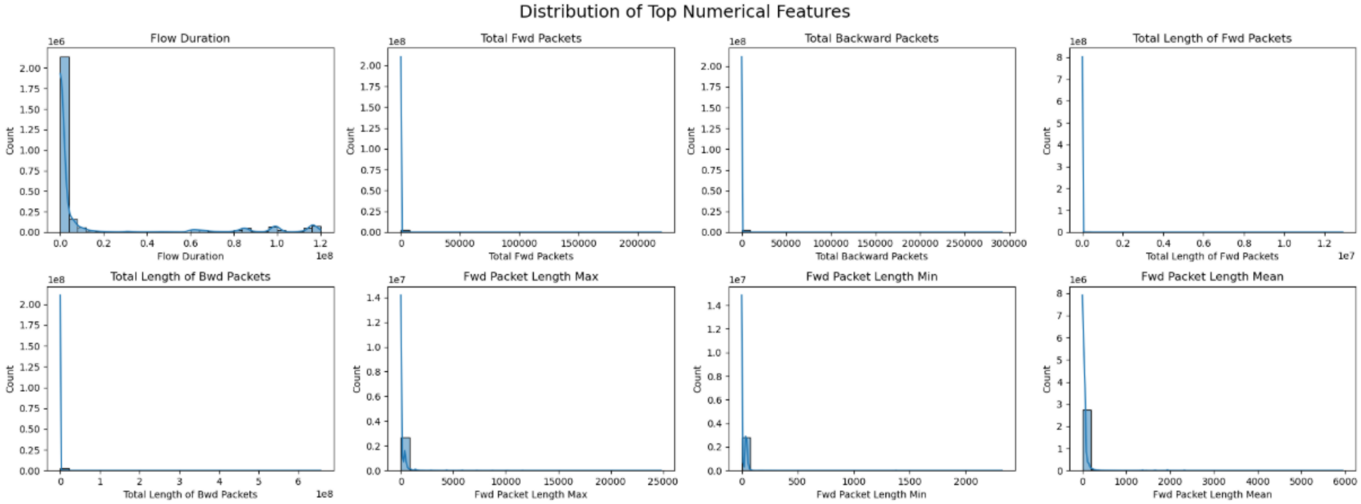
The process of inspecting the dataset then cleaning it, transforming it to a form to find useful information and draw conclusion and support decision making is called Data Analysis. Getting something from raw data to useful information by applying logical and statistical techniques. Data Analysis helps identify the patterns and relationships that can lead to actionable insights and improved performance in various fields.

4.2.1 Data Analysis CICID2017

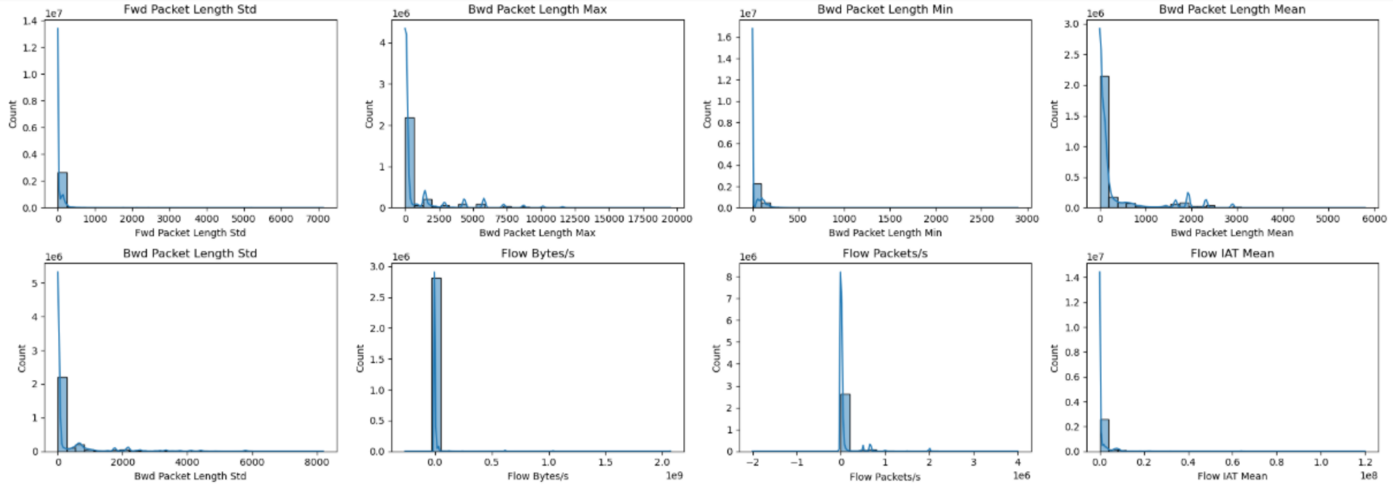
Once the datasets were successfully combined, the next stage involved the analysis of CICID dataset to analyze it so that later we compare the results of it's visualization before and after preprocessing, analyzing CICIDs2017 main dataset.

In this study, data analysis was conducted to understand the patterns, trends, and relationships in the dataset. The dataset name is CICIDS 2017 dataset and is developed by the Canadian Institute for Cybersecurity. The primary objective of the analysis was to show how the data looks visually before the pre-processing of CICID dataset and how dataset look after it's pre-processing. For analysis of main dataset it's important that before applying any preprocessing techniques, an exploratory data analysis (EDA) was performed to gain initial insights, visualize data distributions, and identify potential issues such as missing or inconsistent values. In the analysis of main Dataset of CICID 2017, it includes several steps of code to perform. Python is selected to be the programming language to perform each step of programming including analyzing the dataset. The reason for selecting python programming language is that Python is easy to understand and write and execute. It is a vast language, basically the combination of factors that make it a versatile and accessible programming language. Python is use for various purposes like Web Development, Data Science and Machine Learning, Automation and scripting, Software Development, Scientific Computing and many more.

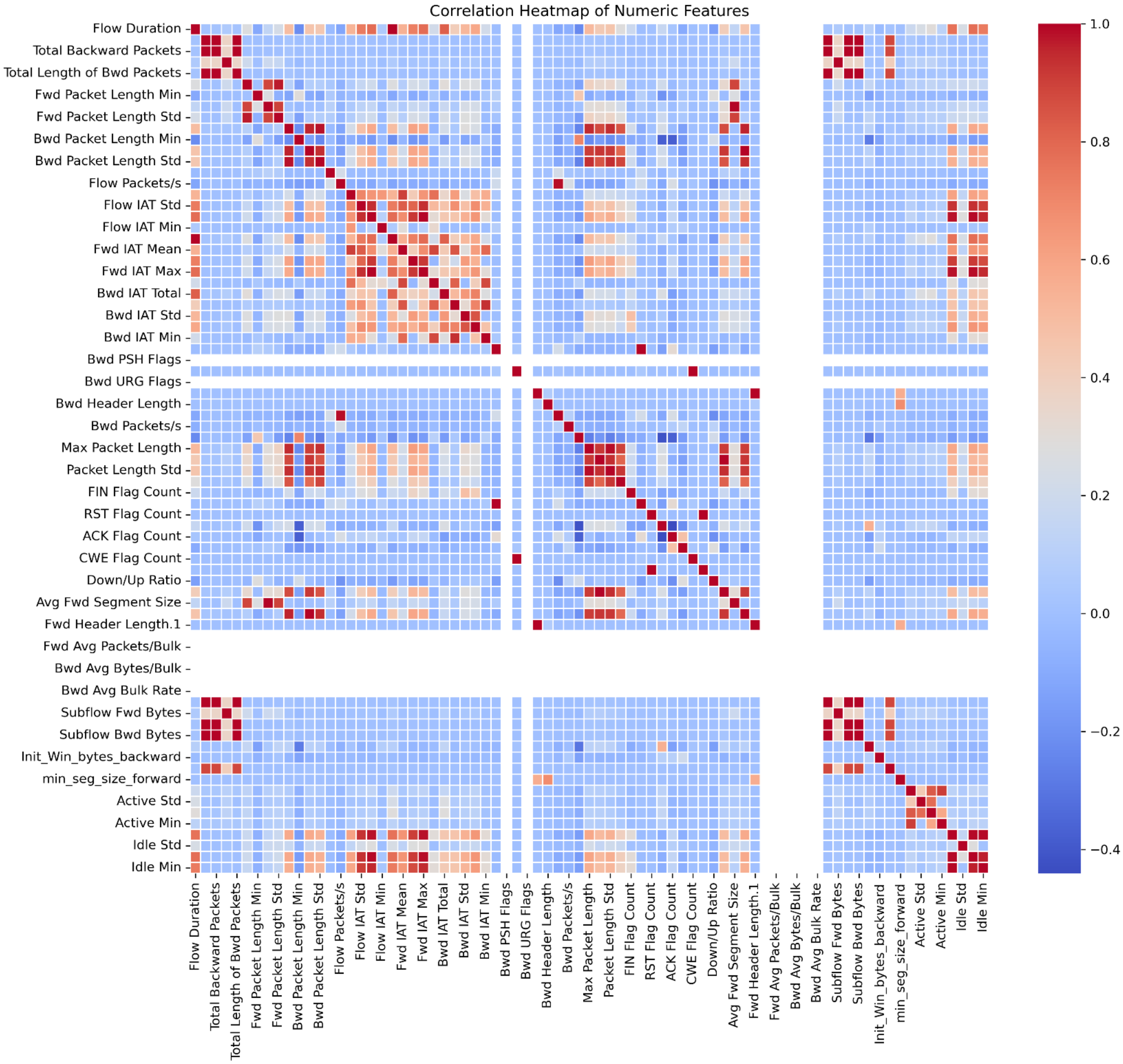
To start the work of Data Analysis, first thing to do is to import necessary libraries used for the work. After importing libraries, dataset is read to understand how to clean the dataset which best fits the AI model, enhancing it's performance. Taking useful features by dropping unnecessary columns and selecting the useful columns. Then it is decided to plot three graphs to analyze the dataset's result before and after it's pre-processing, to observe the difference in dataset after the preprocessing. The first graph to make is a histogram. The result of histograms shows a detailed visual analysis of key numerical features which are extracted from network traffic data. These features are critical in understanding the behavior and structure of data flows within a network, and their distributions provide insight into typical patterns as well as potential anomalies, focuses on metrics such as flow duration, packet counts in both forward and backward directions, length statistics, flow rates, and inter-arrival times and various statistics related to forward packet lengths. These visualizations is the first step to understand network traffic.



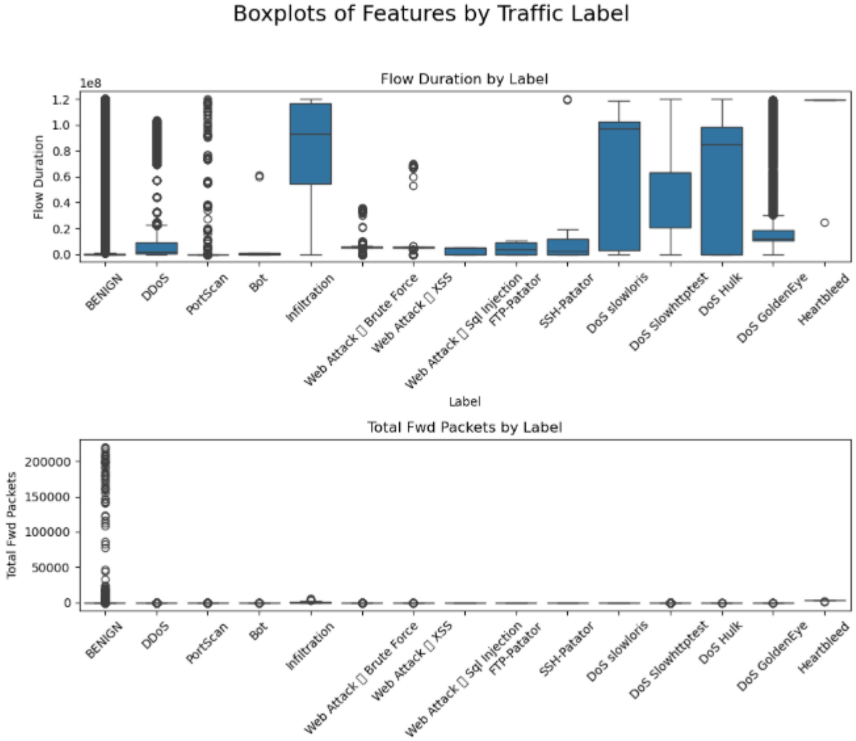
The histograms reveal that the majority of data points for each feature are concentrated at the lower end of the value spectrum. This suggests that most network flows are short-lived and consist of relatively few packets, both in the forward and backward directions. The data of packet length (maximum, minimum, average) skew towards the smaller values, which shows that typical packets are small, but some outliers such as big packets are also shown in the graph. These big packets might be unusual such as suspicious activity or network behaviour.



This analysis explore more packet level and flow level metrics. The standard deviation of forward and backward packet lengths, as well as their respective maximum, minimum, and mean values, again show a strong skew toward lower values. This supports the finding that, although there are occasional exceptions, most packets are small and uniform in size. For evaluating the throughput and timing properties of network traffic, the flow-level metrics—bytes per second, packets per second, and mean inter-arrival time—are very helpful. Additionally, their distributions show scarce occurrences of high-rate flows and a strong concentration at lower rates. These kinds of patterns are common in real-world network settings, where high-volume transfers can occasionally happen but the majority of traffic is modest and sporadic.

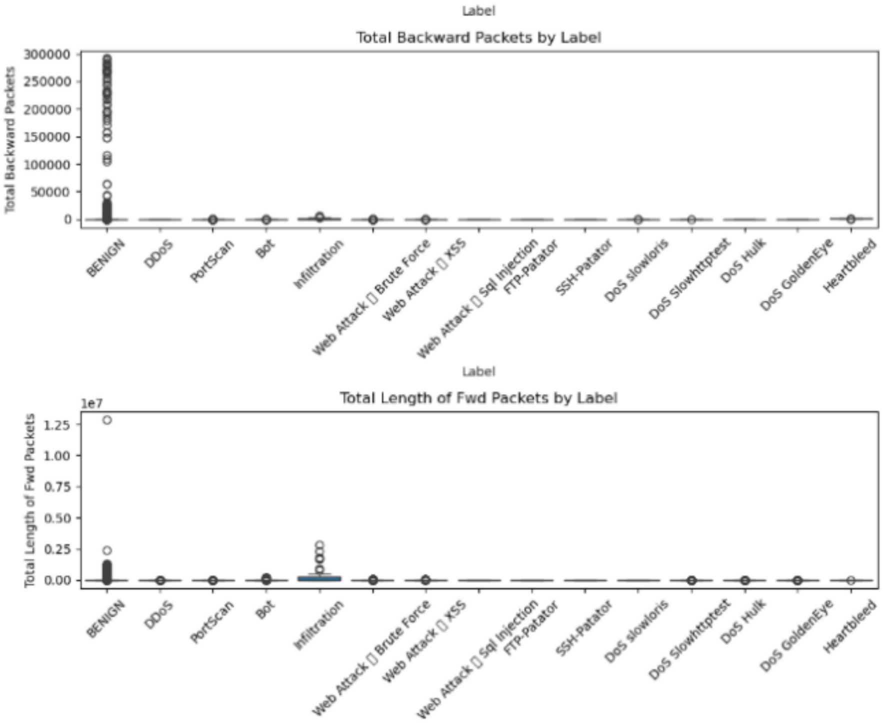


The correlation heatmap shows the relationship between different features of network. If packet size and duration follow similar trend then heatmap shows the strong relationship and the colour become dark there, if the relationship is weak then the colour is light. Researcher use this visualizations to understand the dependencies of variables with each other and there independencies. This is the easy way to visualize and understand the data. The heatmap is structured as a square matrix, with identical variables listed along both the x-axis and y-axis. Each cell within the matrix corresponds to a pairwise correlation coefficient, ranging from -1.0 to 1.0, and is color-coded to reflect the strength and direction of the relationship. A colour gradient is used, with white denoting little to no connection, blue denoting significant negative correlation, and red denoting high positive correlation. The diagonal line of red cells near the center of the heatmap shows how well each variable correlates with itself. This characteristic is common in correlation matrices and provides a visual reference point for understanding the remaining data. Because they show how several variables are related to one another, the off-diagonal cells are very interesting. For example, the heatmap indicates a high positive connection between “Fwd Packet Length Min” and “Fwd Packet Length Std,” indicating that the standard deviation of a forward packet tends to grow as its minimum length does. This would suggest that within some kinds of network flows, packet sizes vary consistently. As would be predicted given their ma“hematical relationshi”, var“ables like "packet len”th Std" and "Packet Length Variance” likewise show a substantial positive association. However, the heatmap also shows negative connections, as the one between “Flow Packets/s” and “Flow Duration.” According to this inverse connection, lengthier flows typically have lower packet rates, which might be a sign of sessions with little activity or idleness. These kinds of insights are useful for differentiating between bursty traffic and streaming traffic, for example. Furthermore, the heatmap’s blue cells indicate regions where variables move in opposing directions, which is useful for spotting repetitive features or anomalies. The heatmap has many different variables, from flow-level features like “Subflow Fwd Bytes” and “Init_Win_bytes_backward” to packet-level metrics like “Bwd Packet Length Min” and “Fwd IAT Std.” Along with flags and header-related characteristics, it includes “Bwd PSH Flags” and “Bwd Header Length,” which are crucial for intrusion detection and protocol analysis. A comprehensive picture of network traffic patterns is provided by the addition of inter-arrival periods and bulk transfer measures, which further enhance the dataset. There are several uses for the correlation heatmap from an analytical perspective. In predictive models, it helps select features by detecting strongly correlated variables that can be redundant or cause multicollinearity. Additionally, it facilitates dimensionality reduction methods by identifying clusters of similar characteristics, such as Principal Component Analysis (PCA). A deeper comprehension of the data’s underlying structure is also made possible by the heatmap, which is essential for creating reliable machine learning models and carrying out efficient network performance evaluations.

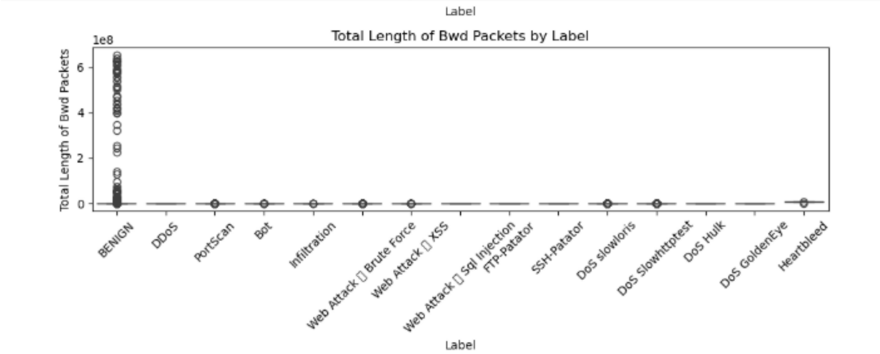


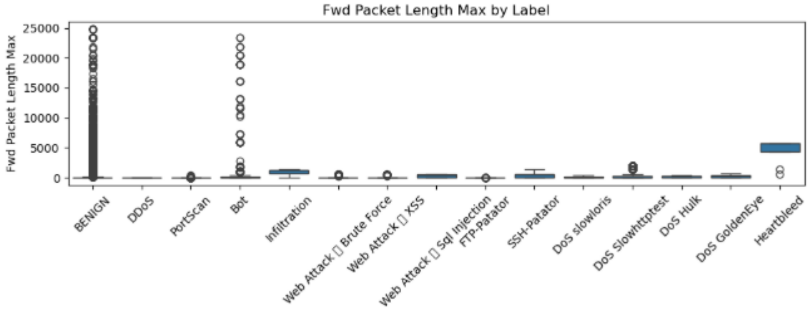
The first figure shows the distributions of Flow Duration and Total Fo’ward Packets by label. The BENIGN and Infiltration labels exhibit a very wide range for Flow Duration, with many outliers, indicating a high variance in how long these traffic flows last. Conversely, other labels like PortScan and DdoS have very short flow durations.

The Heartbleed attack traffic also shows a very long flow duration, similar to Infiltration. For Total Forward Packets, BENIGN traffic shows a wide range and many outliers, suggesting a high variability in packet counts. Other attack types, like DdoS and PortScan, have a much lower number of forward packets.

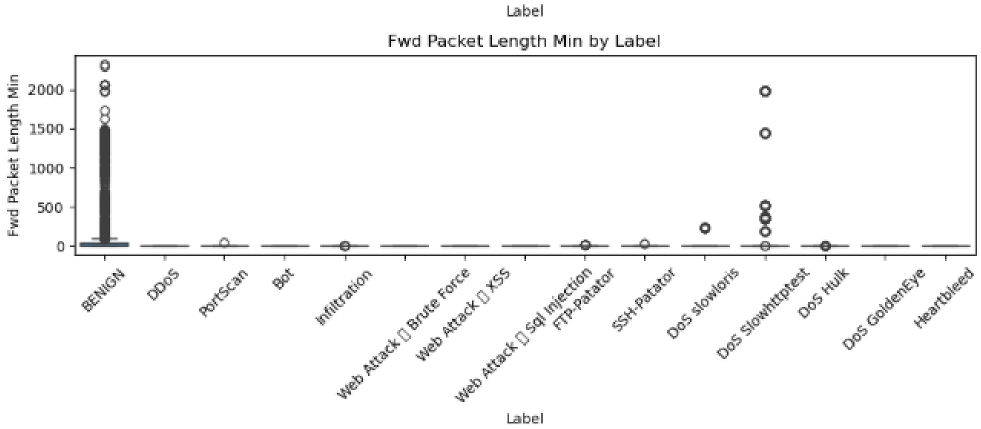


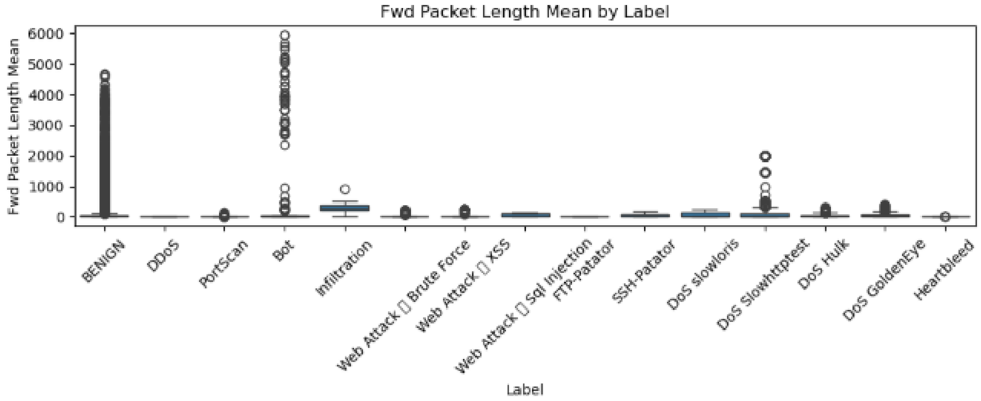
The distributions of the total length of forward packets and the total number of backward packets are shown in the second figure. Similar to the previous plot, BENIGN traffic has the widest distribution and the highest number of outliers for Total Backward Packets, indicating that legitimate traffic has a highly variable number of packets traveling from the destination back to the source. Most attack types, except for Infiltration, have a very low number of backward packets. The Infiltration label shows a slightly higher count of backward packets, which is expected as this type of attack often involves data exfiltration, resulting in some backward traffic. The plot for Total Length of Forward Packets shows that BENIGN and Infiltration traffic have the largest distributions, with many outliers, indicating large amounts of data being sent. Most other attacks show very small packet lengths, with the exception of some outliers for DdoS and PortScan.





The third figure presents the distributions of Total Length of Backward Packets and Fwd Packet Length Max. Again, the BENIGN label has a high variance for Total Length of Backward Packets, with many outliers, indicating that legitimate traffic can have a large total length of backward packets. Most attack types have a total length close to zero. The Fwd Packet Length Max plot reveals that BENIGN, PortScan, and Heartbleed traffic have the highest maximum forward packet lengths, with Heartbleed showing a particularly high maximum length, which is characteristic of the large, malformed TLS packets used in the Heartbleed attack.





The final figure illustrates the distributions of the minimum and mean forward packet lengths. The Fwd Packet Length Min plot shows that the minimum packet length is very low for most traffic types, suggesting that a majority of traffic, both benign and malicious, includes small packets. However, the Fwd Packet Length Mean plot shows a clearer distinction. The BENIGN label has a relatively wide distribution, with a mean packet length ranging from small to moderate. In contrast, labels such as DdoS, Infiltration, and the various DoS attacks have a more constrained and typically lower mean packet length, which can be a key feature for identifying these types of attacks. The Heartbleed label stands out with a very high mean packet length, which again aligns with the nature of the attacks.

4.3 Preprocessing

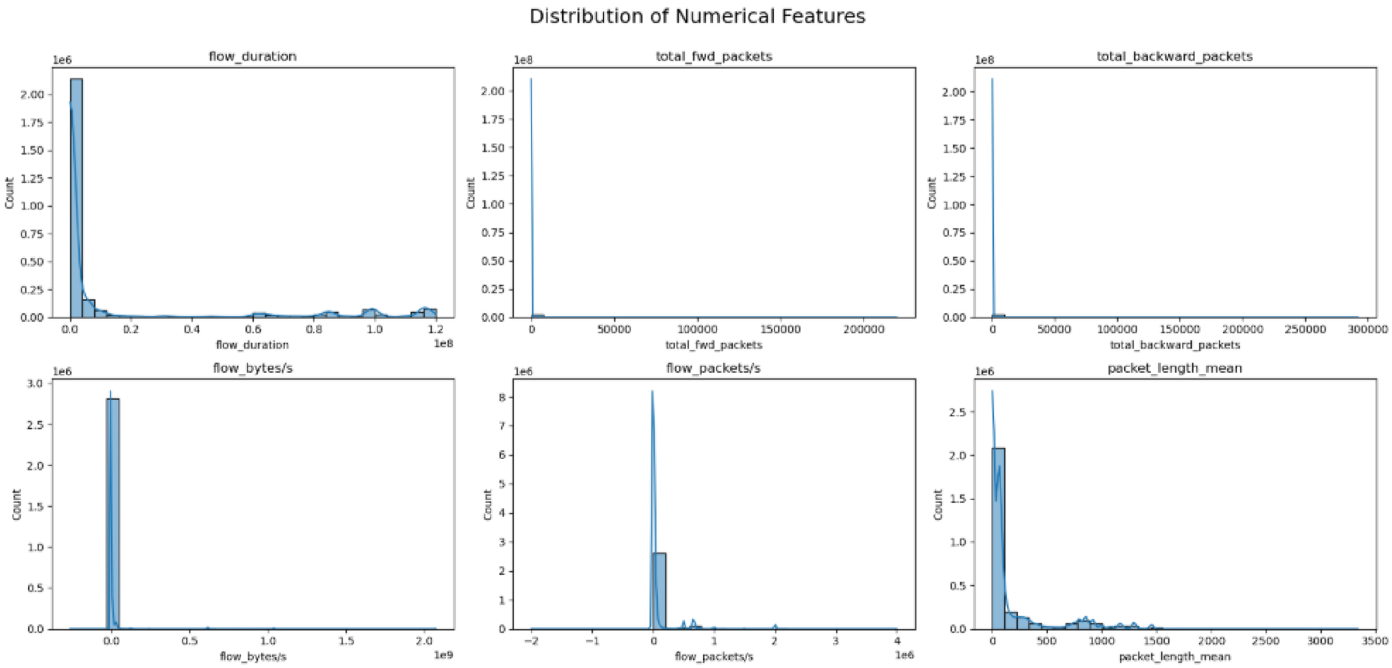
4.3.1 Preprocessing of CICID2017 Dataset

Preprocessing the data to ensure its quality and suitability for model training. This began by importing necessary Python libraries, reading dataset then inspecting the merged dataset for null and infinite values. Null entries were identified using pandas' null value detection capabilities, and subsequently, all rows containing nulls were dropped using the dropna() function to eliminate incomplete records. In order to prevent mistakes and inconsistencies throughout training, this step was crucial. Additionally, the dataset had a large number of infinite values that came from mathematical calculations made during the extraction of traffic features, especially in columns like "flow_bytes/s" and "flow_packets/s." These infinite values were handled by scanning for entries using NumPy's infinity checks and applying a transformation to round all float values whether finite or infinite to a precision of five decimal places. In order to preserve consistency throughout the collection and standardize number representations, this rounding was essential.

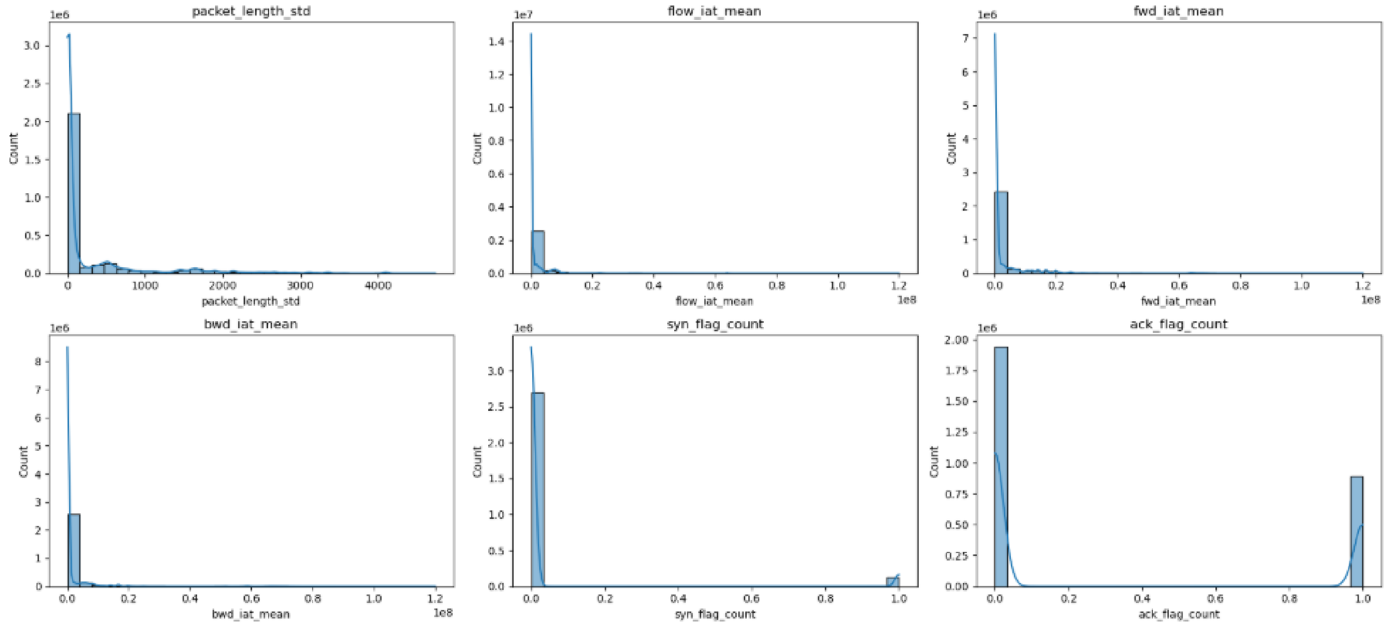
After handling missing and infinite data, the dataset was reduced to only the most relevant features for the classification task. Based on their importance in describing flow-based network characteristics which are particularly crucial in encrypted situations where payload inspection is not feasible a collection of sixteen attributes was chosen. Metrics including flow duration, the total number of packets sent and received, byte rates, packet lengths, inter-arrival periods, TCP flag counts, and behavioral data like active mean duration were among these characteristics. Additionally, the 'label' column was kept as it provided information on whether a certain flow was benign or linked to a particular kind of attack. During model training, this column was the target variable. After the preparation stages were finished, the cleaned and improved dataset was stored in CSV format. This file was used as the last input for the modeling and assessment stages after the pre-processed data was visualized.

4.3.2 Data Analysis of CICID2017 Dataset

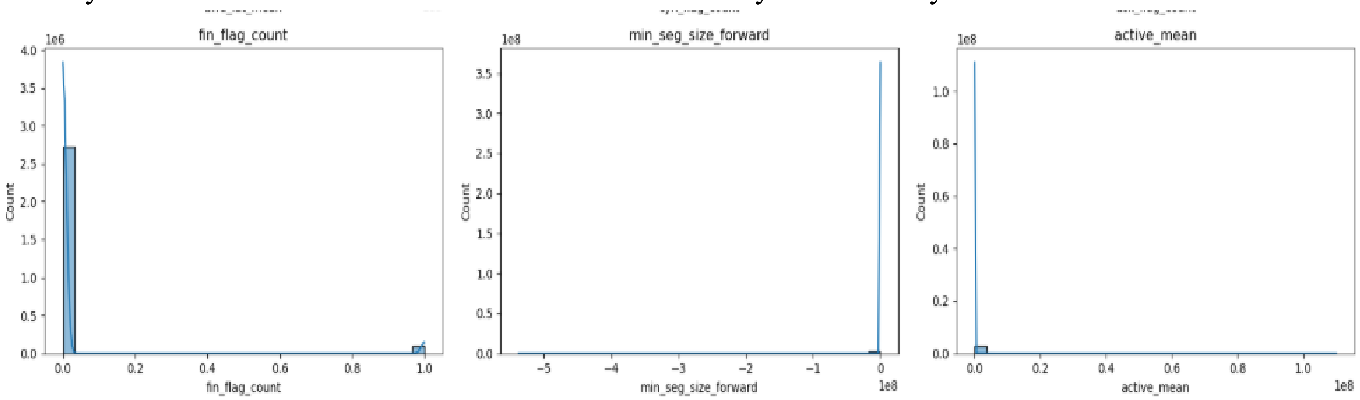
After the preprocessing of dataset it's time to analyze it to see how the dataset looks visually. First visualization to perform is histogram.



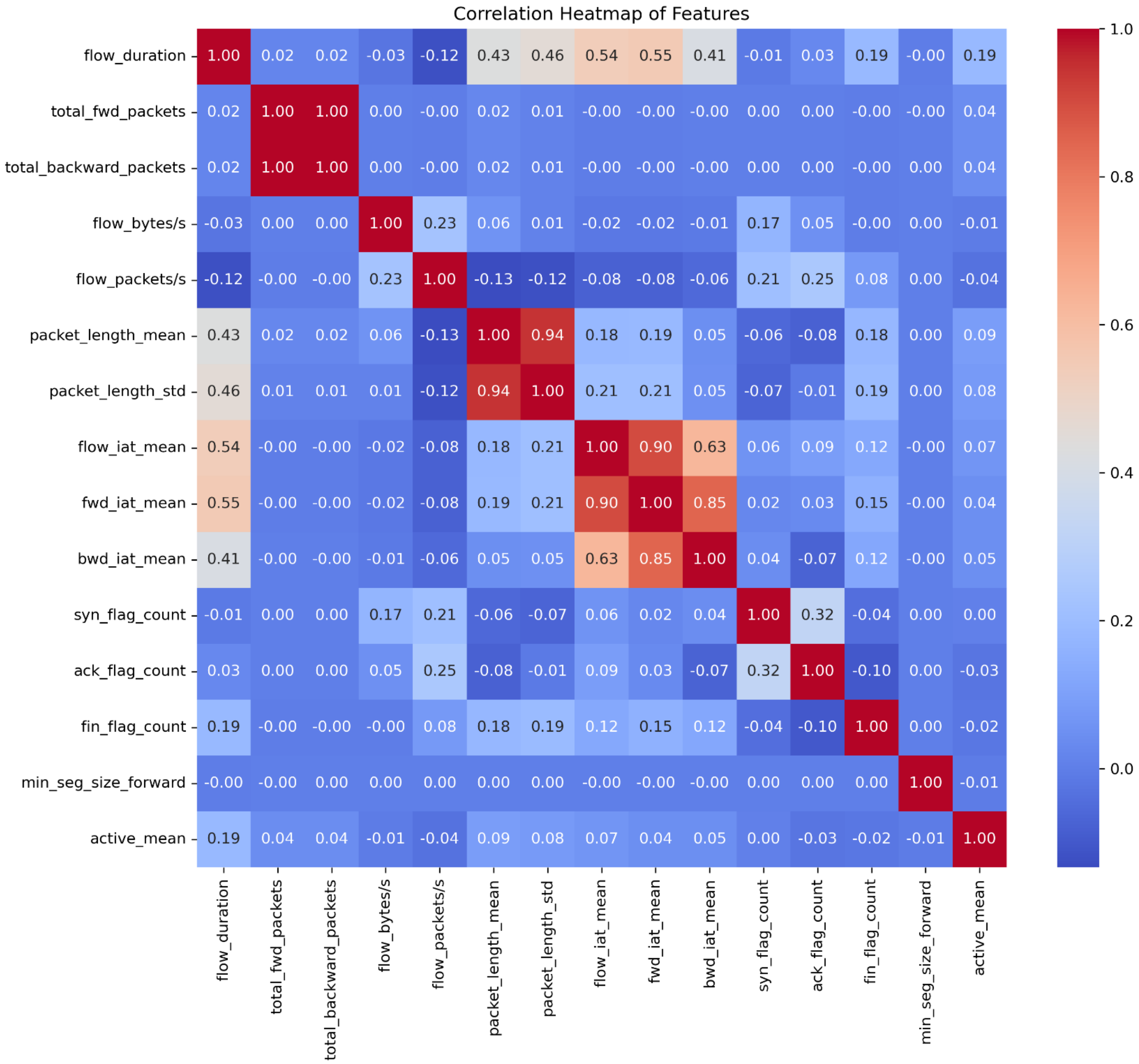
Across the histograms, a repeating pattern of skewed distributions is discernible. Most features, such as flow_duration, total_fwd_packets, total_backward_packets, flow_bytes/s, and packet_length_mean, are heavily skewed to the right. This indicates that the majority of data points have values close to zero, with a long tail extending to the right, representing a smaller number of instances with very high values. Network traffic data frequently exhibits this kind of distribution, with the majority of flows being brief and containing few packets.



While a few flows are exceptionally long or contain a large number of packets. The features packet_length_std, flow_iat_mean, fwd_iat_mean, and bwd_iat_mean also exhibit this right-skewed behavior. On the other hand, certain traits exhibit discrete, non-continuous distributions. The majority of counts are grouped at zero, with a noticeable spike at one, according to the syn_flag_count and fin_flag_count histograms. Accordingly, these attributes are probably binary or indicate the frequency of a certain occurrence (for example, the presence of a SYN or FIN flag once in a flow). With a sizable portion of flows having an ack_flag_count of zero and another sizable group having an ack_flag_count larger than zero, ack_flag_count also exhibits a bimodal distribution, which may be an indication of the distinction between one-way and two-way communication.



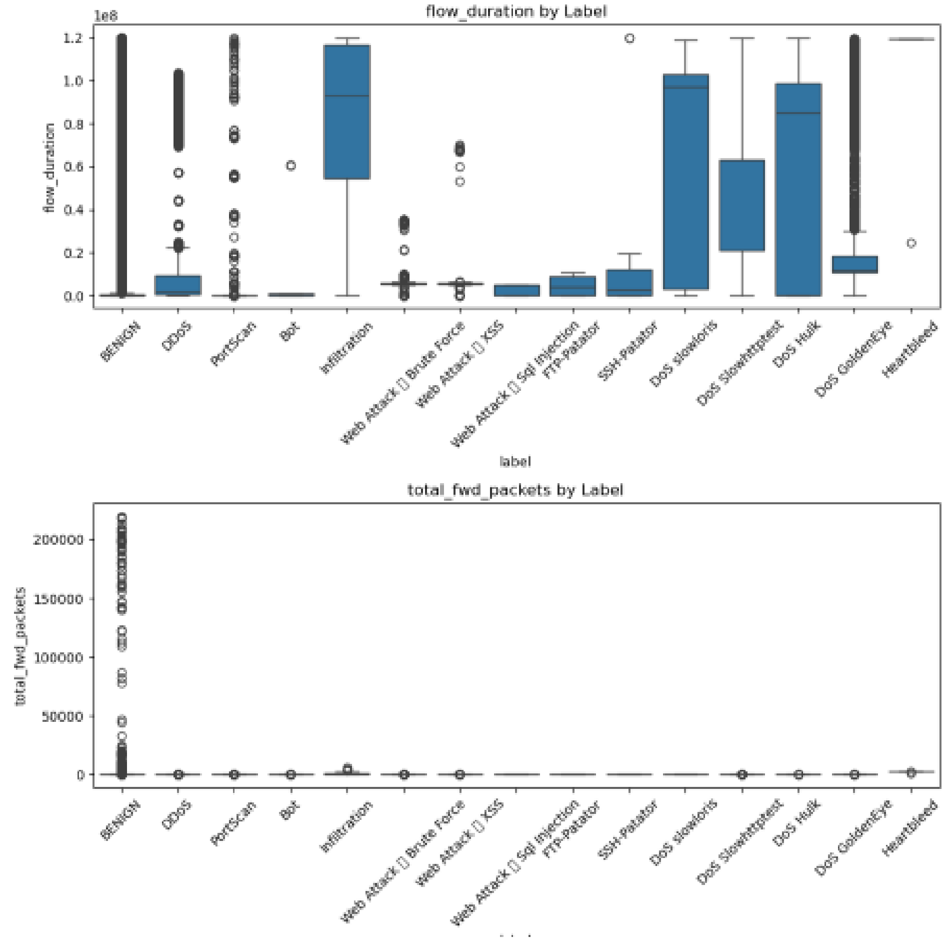
After pre-processing, some histograms, such min_seq_size_forward, seem to have a single, abrupt spike, suggesting that the data for that specific feature is not very variable. This suggests that nearly all data points for min_seq_size_forward have the same value, possibly zero, after some form of transformation or normalization. With the majority of values clustered close to zero, the characteristic active_mean also exhibits a severely skewed, nearly single distribution. Overall, the research shows that in order to manage the high skewness and broad range of values, which may otherwise impair the performance of machine learning models, careful feature scaling and transformation are required.



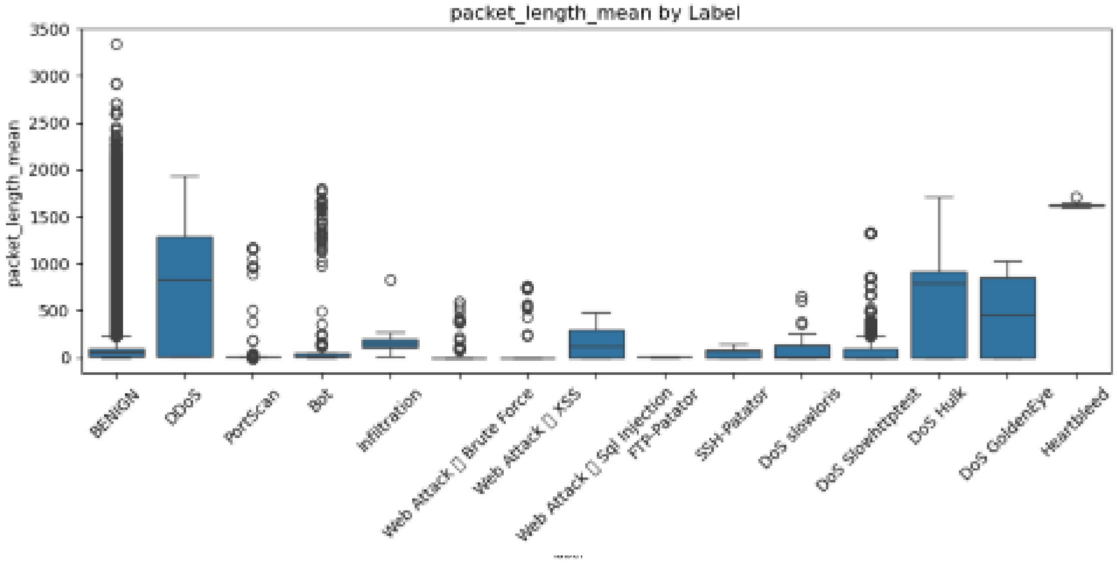
Now it's time to plot a detailed correlation heatmap which meticulously displays the pairwise linear relationships between various numerical features of the dataset. A statistical metric that assesses the strength and direction of a linear relationship between two variables, the Pearson correlation coefficient is represented by each cell in this grid. Its value ranges from -1 to +1. A perfect positive correlation (+1.00) is represented by a deep red color bar, a perfect negative correlation (-1.00) by a deep blue color bar, and no linear correlation (0.00) by a white or light gray color bar on the right. The diagonal, from the top-left to the bottom-right, is filled with values of 1.00, as any feature is perfectly correlated with itself. A striking observation is the perfect positive correlation of 1.00 between total_fwd_packets and total_backward_packets, which strongly suggests that in this dataset, the number of packets sent in one direction is perfectly proportional to the number sent in the other. This indicates that these two features are essentially providing redundant information and one of them could be removed for model training to avoid the issue of multicollinearity. Similarly, the inter-arrival time metrics are highly correlated with each other; for example, flow_iat_mean shows a strong positive correlation of 0.90 with fwd_iat_mean and 0.85 with bwd_iat_mean, which is logical as these features all measure aspects of packet timing. A strong positive correlation of 0.94 is also evident between packet_length_mean and packet_length_std, implying that network flows with a larger average packet size also tend to have a wider variance in their packet lengths.

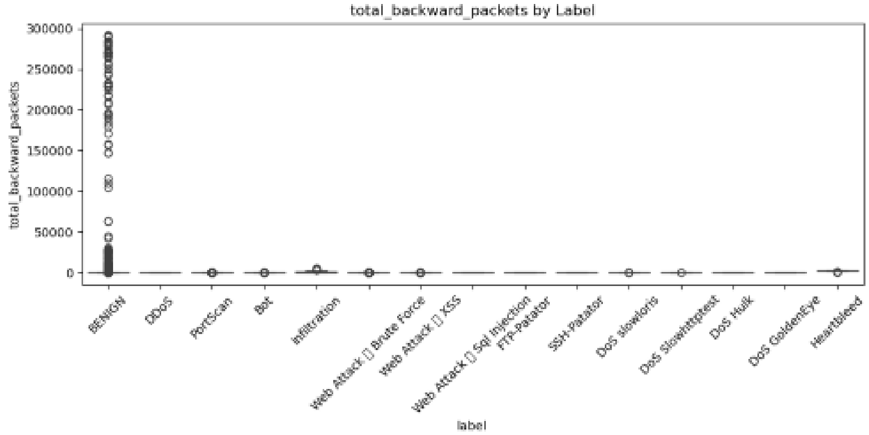
Significant but less severe correlations are also revealed by the heatmap. For example, flow_duration has a somewhat positive correlation with flow_iat_mean (0.54), fwd_iat_mean (0.55), and bwd_iat_mean (0.41), suggesting that longer flows often have bigger average inter-arrival periods between packets. The heatmap further shows that flow_bytes/s has very low correlations with most other features, suggesting it might capture unique information not present in the other variables. Knowing these linkages directly affects machine learning, therefore it's not simply an intellectual exercise. When choosing features, it is important to take large correlations into account because doing so may result in models that are unstable and difficult to understand. Therefore, this heatmap is a vital tool for the exploratory data analysis stage, helping to decide which variables to retain, modify, or exclude prior to training a predictive model.

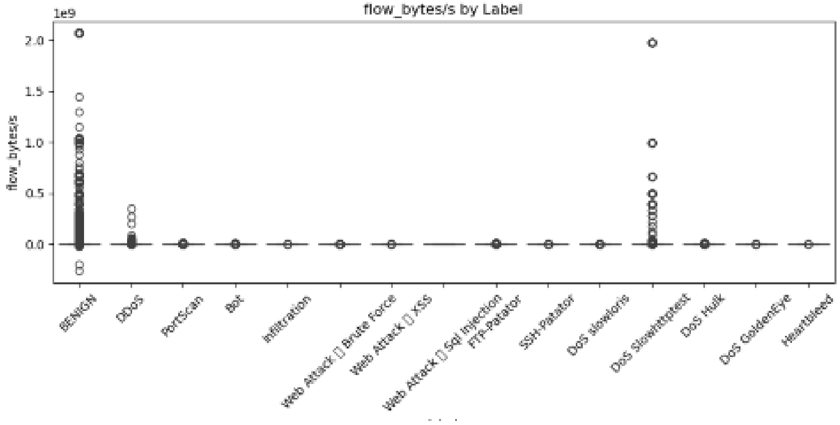
A sequence of box plots that show the distribution of many numerical characteristics across various traffic labels—from normal ("BENIGN") to different kinds of attacks—are displayed in the visualization. After examining correlations, these plots are an essential step in comprehending how feature distributions differ by class, which is essential for creating a strong classification model. The median (the line within the box), the first and third quartiles (the box's sides), and the minimum and maximum values (the whiskers' ends) are the five essential statistics that each box plot uses to summarize the distribution of a feature. Potential outliers are shown by points that extend past the whiskers.



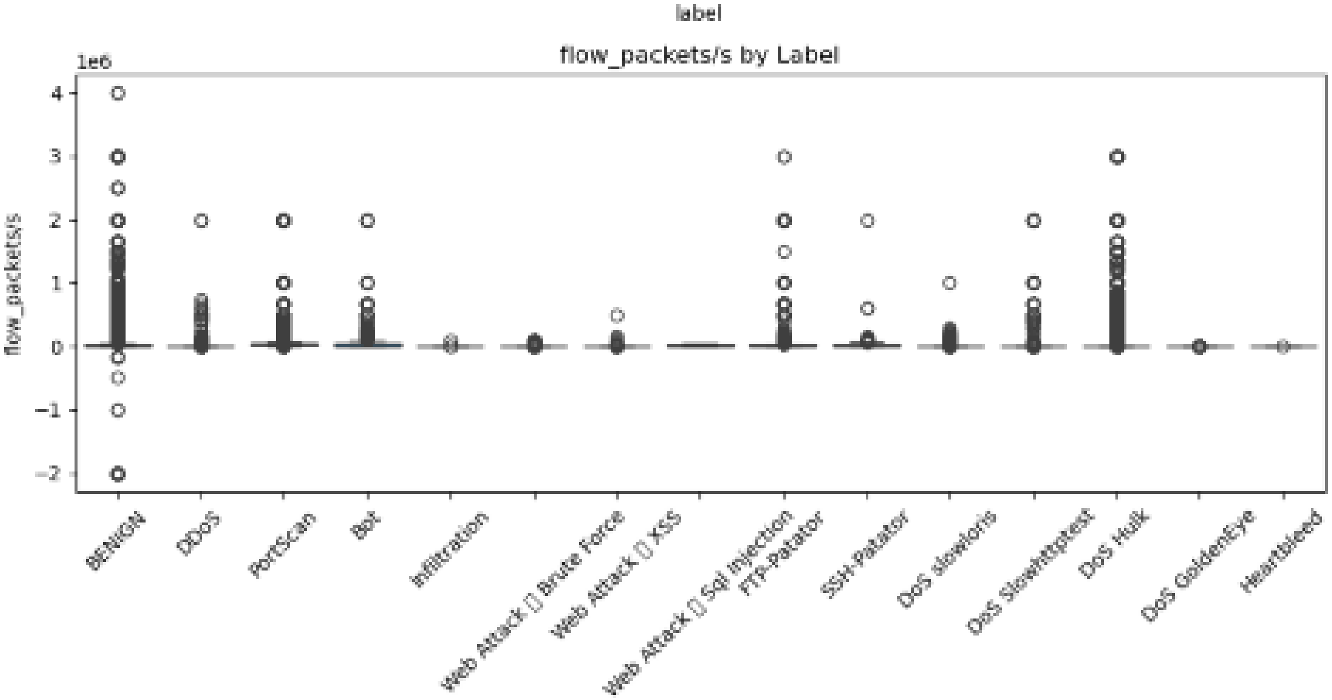
The box plots for flow_duration and total_fwd_packets show significant differences across labels. For flow_duration, benign traffic and some attack types like PortScan and DoS attacks exhibit a wide range of values, with some attack categories such as PortScan, Infiltration, and DoS slowloris showing a particularly high median duration. A large degree of diversity in flow durations is indicated by the existence of many outliers, particularly in benign traffic. Similarly, the total_fwd_packets and total_backward_packets box plots show that benign traffic has a very wide distribution, with a high concentration of flows having a small number of packets but a large number of outliers with very high packet counts. In contrast, many attack types, such as **DdoS** and **PortScan**, have a much tighter distribution for these features, with most flows containing a small number of packets and a low median value. This suggests that the total number of packets could be a distinguishing feature between benign and certain attack types.



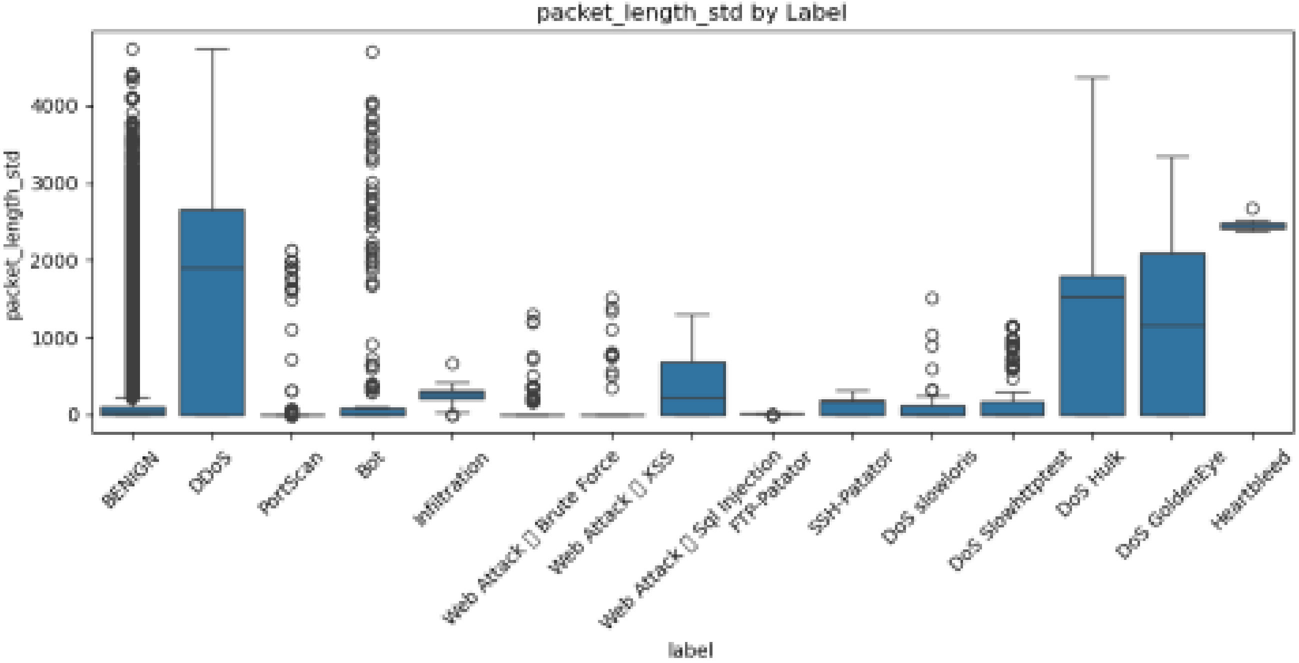




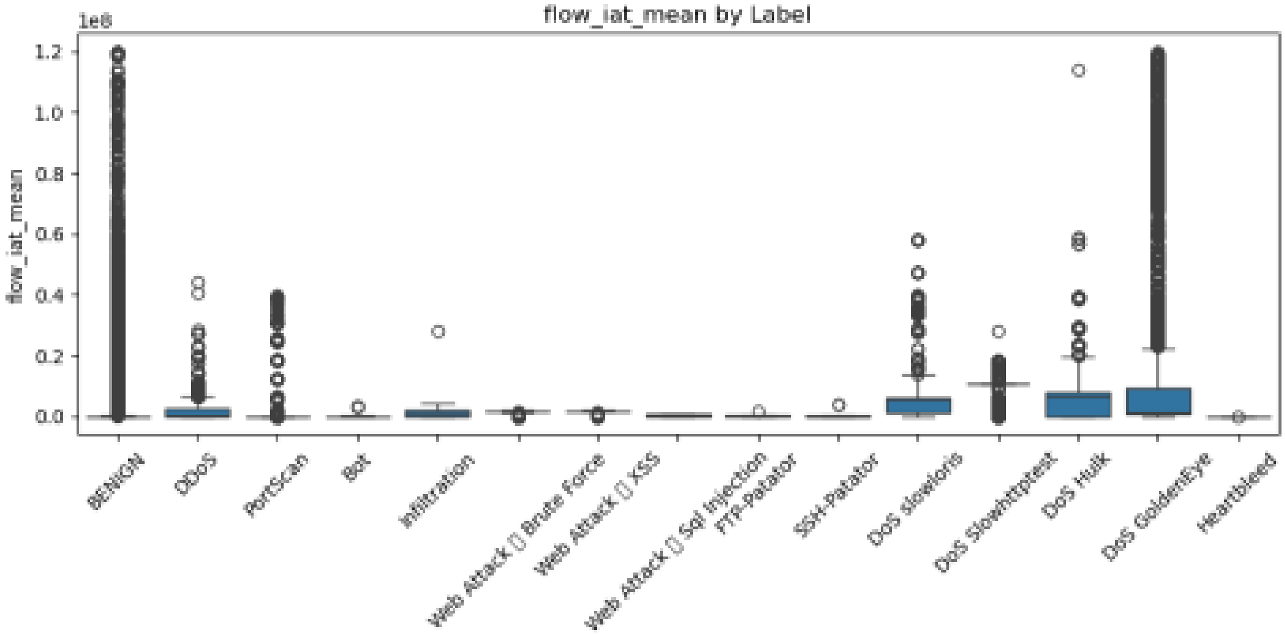
The box plots for flow_bytes/s and packet_length_mean also reveal interesting class-specific behaviors. The flow_bytes/s graphic reveals that although the median for the majority of labels is near zero, there are notable outliers that indicate flows with exceptionally high throughput, especially in benign traffic and some assaults. The packet_length_mean box plot shows a clear distinction, where certain attacks like **DoS Hulk** and **DoS GoldenEye** have a significantly higher median packet length compared to benign traffic and other attacks. This implies that a good way to spot these particular kinds of assaults could be to look at the average packet size.



The "flow_packets/s by labels" box plot illustrates the distribution of the flow_packets/s feature, which represents the number of packets per second for a given network flow, across different traffic labels. The pace of packet transmission for benign traffic is efficiently contrasted with different attack types in this picture. The plot shows that for most labels, including benign traffic, the median value is very close to zero, suggesting that a significant portion of flows have a low packet rate. However, the presence of numerous outliers, particularly for benign traffic, indicates that while most benign flows have a low packet rate, a small number of them have exceptionally high rates. Conversely, most attack types, such as **DdoS** and **PortScan**, show a much tighter distribution with fewer high-value outliers, suggesting these attacks are characterized by a more consistent, and often low, packet per second rate. The potential of flow_packets/s as a differentiating characteristic for recognizing and categorizing various kinds of network traffic is demonstrated by its clear distributional pattern.



packet_length_std by Label



flow_iat_mean by Label

Furthermore, the box plots for packet_length_std and flow_iat_mean underscore the variability within and between classes. For packet_length_std, DoS attacks like **DoS Hulk** and **DoS GoldenEye** again show a higher median and a wider distribution, implying greater variance in packet sizes within these flows. For benign traffic, the flow_iat_mean figure has a relatively high median and broad dispersion; in contrast, many attack categories have significantly lower medians, which indicate quicker packet rates. Numerous outliers are present in all characteristics and labels, highlighting the necessity of treating these values robustly. These outliers may be a sign of various attack subtypes or typical but unexpected network activity. The distinct distributions observed for several features strongly suggest that these variables hold significant predictive power for classifying different types of network traffic and attacks.

4.3.3 Heatmap comparison: Raw vs Preprocessed CICID2017

Upon comparison of analysis of the two correlation heatmaps, one representing the raw dataset and the other depicting the same dataset after preprocessing, several key differences emerge. With a huge number of characteristics, the first heatmap, which displays the raw data, is much bigger and more intricate. A high degree of multicollinearity among the initial features is indicated by the wide grid of correlations, many of which have strong positive or negative values, making this complexity clearly obvious. For example, many feature groups, including those associated with byte counts (Bwd Header Length, Bwd Packet/Bulk), inter-arrival times (Flow IAT Std, Flow IAT Mean, Flow IAT Total), and packet length (Packet Length Min, Packet Length Max, and Packet Length Std), show very strong correlations with each other. This suggests that the raw dataset contains a lot of redundant information, where multiple features are capturing similar underlying aspects of the network traffic. It is challenging to rapidly determine the most crucial variables or 32reatee a unwilling model due to the overwhelming number of data and their complex interactions.

The second heatmap, which was produced following preprocessing, is in sharp contrast, far more straightforward and targeted. This change is the outcome of a purposeful feature selection and engineering procedure that eliminated or consolidated duplicate or less instructive aspects. The preprocessed heatmap displays a much smaller, more manageable set of features. The overall complexity has been significantly decreased, even if some strong correlations still exist, such as the perfect positive correlation between total_fwd_packets and total_backward_packets. In general, the relationships among the remaining traits are less overwhelming and more clear. For example, the strong correlation between flow_duration and flow_iat_mean (0.54) and fwd_iat_mean (0.55) is now more prominent and easier to interpret in the context of the smaller feature set. The most important results of the preprocessing step are the obvious decrease in the number of characteristics and the simplicity of the correlation matrix. By reducing problems like multicollinearity and high dimensionality that were so common in the raw data, this procedure not only makes the data easier to examine and visualize but also lays the groundwork for creating more effective and reliable machine learning models.

4.4 Training of Models

Following the successful preprocessing and Analysis of the CICIDS2017 dataset, the next critical phase in the methodology involved training a range of machine learning and deep learning models to detect hidden cyber threats in encrypted network data. In addition to evaluating model accuracy, this phase compared the performance, generalization, and appropriateness of more sophisticated neural networks for intrusion detection systems with more conventional approaches.

4.4.1 Deep Learning Models Training on CICID2017 Dataset

The implementation of a one-dimensional convolutional neural network (1D-CNN) architecture marked the start of the deep learning efforts. The preprocessed CICIDS2017 dataset was used to train this model, which showed a test accuracy of roughly 93.18%. Both benign and malicious traffic achieved scores of approximately 0.93, according to the classification report, which showed a balanced performance between precision and recall. Building on this, a two-dimensional convolutional neural network (2D-CNN) was used, resulting in a significantly increased generalization and an overall test accuracy of 94.28%. An Elman Recurrent Neural Network (ERNN), which achieved a test accuracy of 95.71% after three epochs, with low loss values and improved f1-scores, was used to further explore recurrent architectures. A test accuracy of 93.35% was attained by the Gated Recurrent Unit (GRU) model, which was also trained over three epochs. A Long Short-Term

Memory (LSTM) network was also evaluated, and it performed well in identifying sequential dependencies in the data, with a test accuracy of 93.90%. The Residual Neural Network (ResNet), which was the product of the deep learning phase, attained a constant accuracy of 91% on the test set in spite of its longer training duration and increased computing load. ResNet performed well, however neither raw accuracy nor classification balance allowed it to outperform the ERNN.

4.4.1.1 Result of all Deep Learning models on CICID2017

Datasets	Deep Learning Models	Accuracy	Recall	F1-Score	Precision	Confusion Matrix
CICID 2017	1D CNN	93.18%	0.93	0.93	0.93	[[103610 7702] [7468 103843]]
CICID 2017	2D CNN	94.28%	0.95	0.95	0.95	
CICID 2017	ERNN	95.71%				
CICID 2017	GRU	93.35%				
CICID 2017	Long Short-Term Memory (LSTM)	93.90%				
CICID 2017	ResNet	90.06%	0.91	0.91	0.91	[[107051 4350] [16367 94855]]

4.4.2 Machine Learning Models training on CICID2017 Dataset

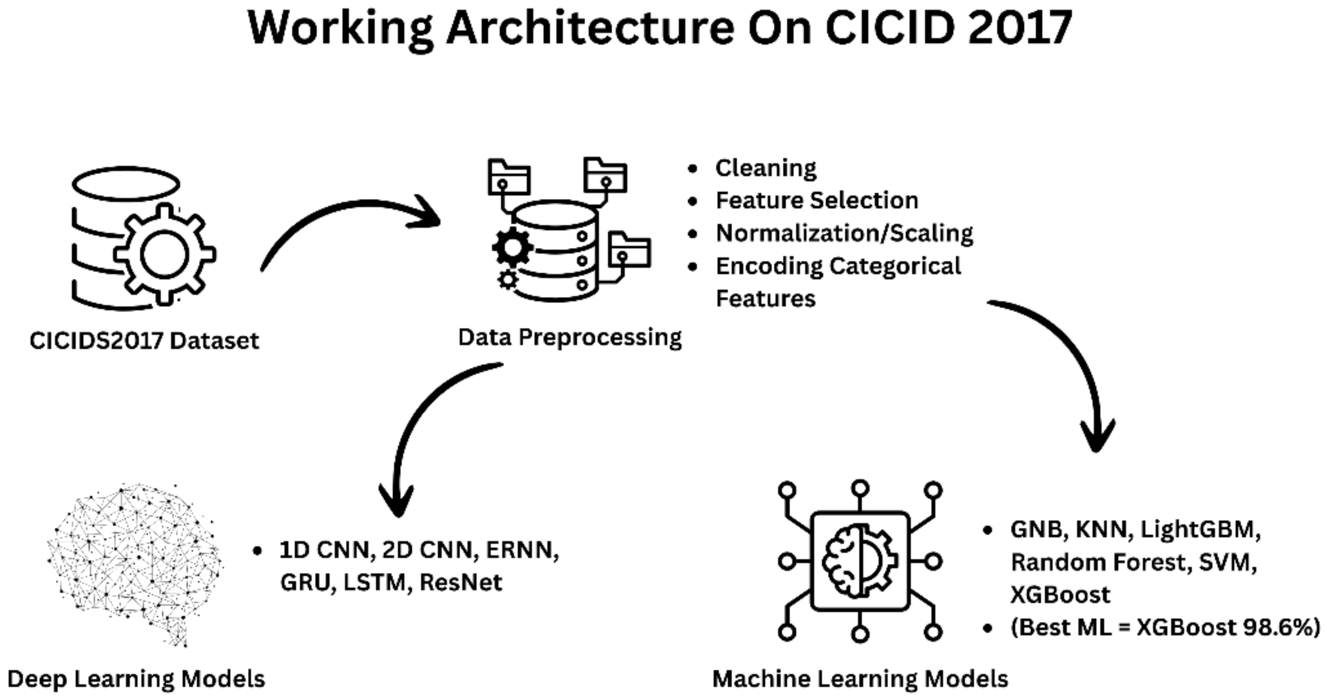
Parallel to deep learning experiments, several machine learning algorithms were also trained and evaluated on the CICIDS2017 dataset. The Naïve Bayes classifier (GNB) performed poorly, having a very low recall for benign traffic and an overall accuracy of only 50.13%. The Light Gradient Boosting Machine (LightGBM) considerably outperformed conventional machine learning models, attaining a high test accuracy of 98.60%, while the K-Nearest Neighbors (KNN) model showed good generalization. Excellent classification metrics in all classes and consistent cross-validation scores bolstered LightGBM's performance. With an accuracy of 98.38%, Random Forest trailed closely behind, demonstrating resilience in managing the high-dimensional feature space of the dataset. With a respectable accuracy of 92.13%, the Support Vector Machine (SVM) trailed slightly behind ensemble-based methods. Lastly, with a test accuracy of 98.62%, the Extreme Gradient Boosting (XGBoost) model outperformed LightGBM, confirming the effectiveness of gradient-boosting algorithms for network intrusion detection.

4.4.2.1 Result of all Machine Learning models on CICID2017

Datasets	Machine Learning Models	Accuracy	Recall	F1-Score	Precision	Confusion Matrix
CICID 2017	Random Forest Classifier	98.38%	0.98	0.98	0.98	[[9709 203] [121 9967]]
CICID 2017	Support Vector Machine (SVM)	92.13%	0.92	0.92	0.92	[[3714 256] [374 3656]]
CICID 2017	K- Neighbors Classifier	98.53%	0.99	0.99	0.99	[[109465 1936] [1339 109883]]
CICID 2017	Guassian Naïve Bayes	50.13%	0.5	0.34	0.7	[[517 166450] [64 166903]]

CICID 2017	Light GBM	98.60%	0.99	0.99	0.99	[[109805 1596] [1512 109710]]
CICID 2017	XGBoost	98.62%	0.99	0.99	0.99	[[109824 1577] [1495 109727]]

4.4.3 Architecture of Models training on CICID2017 Dataset


Working Architecture On CICID 2017
CICIDS2017 Dataset
Data Preprocessing
• Cleaning
• Feature Selection
• Normalization/Scaling
• Encoding Categorical Features
Deep Learning Models
• 1D CNN, 2D CNN, ERNN, GRU, LSTM, ResNet
Machine Learning Models
• GNB, KNN, LightGBM, Random Forest, SVM, XGBoost
• (Best ML = XGBoost 98.6%)

The architecture above shows that CICID2017 dataset is preprocessed using techniques and the preprocessed data is used in the training of Machine Learning and Deep Learning models.

4.4.4 Machine Learning models training on UNSW-NB15

The UNSW-NB15 Dataset is the next dataset to be trained after the CICIDS2017 dataset was finished. Using the UNSW-NB15 dataset, this study assesses several machine learning and deep learning models for network intrusion detection. The four dataset components were first combined into a single file and pre-processed by substituting the mean for missing and infinite values. For models like KNN and SVM that are sensitive to input magnitude, feature scaling was implemented using Standard Scaler. The accuracy of the K-Nearest Neighbors (KNN) model was 99.22% after normalization. Gradient boosting models like LightGBM and XGBoost were trained after imputation and cross-validation; LightGBM achieved an accuracy of 98.82%, while XGBoost achieved an accuracy of 99.04%. Random Forest, another ensemble model, also attained a high accuracy of 99.22%.

4.4.4.1 Result of all Machine Learning models on UNSW-NB15

Datasets	Machine Learning Models	Accuracy	Recall	F1-Score	Precision	Confusion Matrix
UNSW-NB15	Guassian Naïve Bayes	81.69%	0.82	0.81	0.86	[[41653 22768] [768 63325]]
UNSW-NB15	K- Neighbors Classifier	99.04%	0.99	0.99	0.99	[[63471 950] [286 63807]]
UNSW-NB15	Light GBM	99.25%	0.99	0.99	0.99	[[63499 922] [40 64053]]
UNSW-NB15	Random Forest Classifier	99.23%	0.99	0.99	0.99	[[63568 853] [142 63951]]
UNSW-NB15	Support Vector Machine (SVM)	95.45%	0.96	0.95	0.96	[[960 75] [16 949]]
UNSW-NB15	XGBoost	99.25%	0.99	0.99	0.99	[[63525 896] [65 64028]]

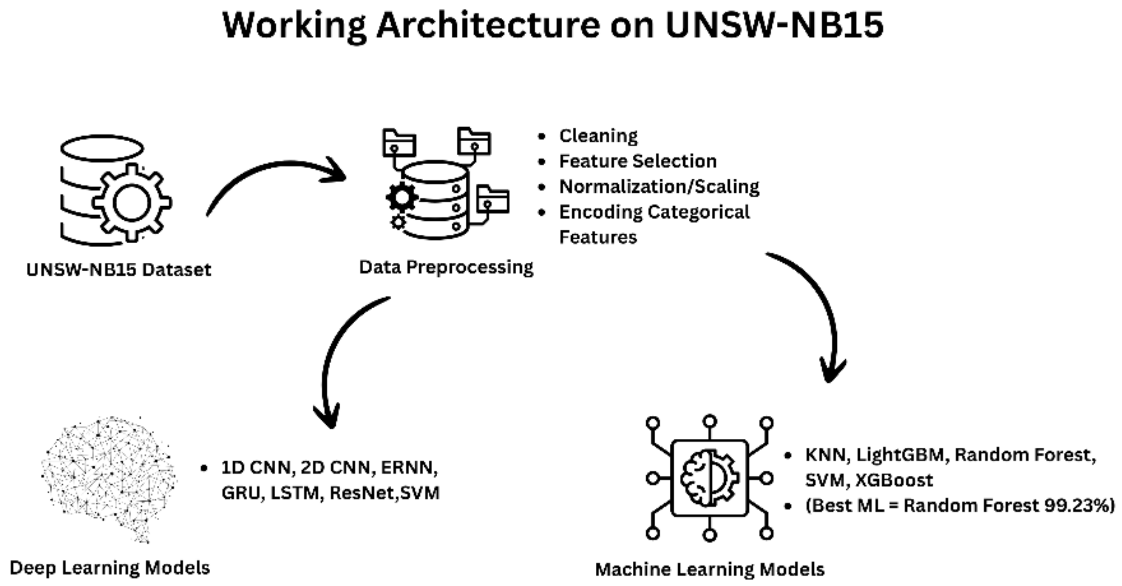
4.4.5 Deep Learning models training on UNSW-NB15

Deep learning models including LSTM, GRU, and Elman RNN had to convert features into 3D input for sequential learning, and they achieved an accuracy of 87.37%. Convolutional models were also employed: 1D-CNN, which employed reshaped 3D input for temporal filtering, achieved 98.55%, while 2D-CNN, which employed grid-like 2D reshaping, acquired 95.65%. The Gaussian I Bayes model is a simple probabilistic classifier that has an accuracy of 81.75% after imputation. The Support Vector Machine (SVM) classifier was then trained using the scaled and cleaned data, and it produced an accuracy of 97.02%. The UNSW-NB15 dataset allows for the consistent and comparable evaluation of different models, highlighting the superior performance of ensemble and distance-based methods.

4.4.5.1 Result of all Deep Learning models on UNSW-NB15

Datasets	Machine Learning Models	Accuracy	Recall	F1-Score	Precision	Confusion Matrix
UNSW-NB15	1D CNN	98.55%				
UNSW-NB15	2D CNN	95.64%				
UNSW-NB15	ERNN	87.36%				
UNSW-NB15	GRU	87.36%				
UNSW-NB15	LSTM	87.36%				

4.4.6 Architecture of Models training on UNSW-NB15 Dataset


Working Architecture on UNSW-NB15
UNSW-NB15 Dataset
Data Preprocessing
• Cleaning
• Feature Selection
• Normalization/Scaling
• Encoding Categorical Features
Deep Learning Models
• 1D CNN, 2D CNN, ERNN, GRU, LSTM, ResNet,SVM
Machine Learning Models
• KNN, LightGBM, Random Forest, SVM, XGBoost
• (Best ML = Random Forest 99.23%)

This architecture shows that UNSW-NB15 dataset is collected and preprocessed using preprocessing techniques and the preprocessed data is used in the training of Machine Learning and Deep Learning models.

4.4.7 Discussion after Training Models

The findings from both datasets showed that ensemble learning models, in particular LightGBM and XGBoost, performed the best in identifying concealed cyberthreats in encrypted data in the CICID 2017 dataset. Meanwhile Random Forest Classifier performing the best in UNSW-NB15 dataset. With strong precision, recall, and f1-score values, XGBoost outperformed all other models tested, attaining the maximum accuracy on CICIDS2017 and Random Forest emerged as the best-performing model on UNSW-NB15 dataset. In terms of accuracy and generalization, ERNN also outperformed CNNs, LSTM, GRU, and ResNet, exhibiting the best performance among deep learning models. According to these results, gradient boosting and recurrent learning methods are effective in creating reliable intrusion detection systems that can function in encrypted settings.

4.5 Hybrid Model

A hybrid model is a type of model where two or more than two models, approaches or techniques combined together to form a new model so that better performance and better accuracy can be achieved.

4.5.1 Autoencoder

A neural Network which compress the input data and then reconstructs the same data so that it learn important features. Autoencoder is an unsupervised learning technique which learn the hidden patterns of data. It’s purpose is to learn important and useful features from the data automatically and remove extra noise from the data.
Using the CICIDS 2017 dataset, this study uses a hybrid machine learning approach to identify intrusions in network traffic after implementing machine learning and deep learning models. The suggested approach incorporates an XGBoost classifier to carry out the final classification after a deep learning-based autoencoder for feature extraction and dimensionality reduction. The reason for using XGBoost is due to its best performance.

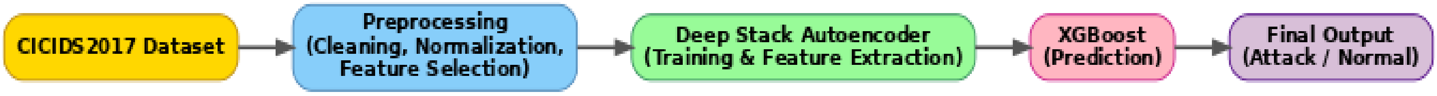
The entire pipeline ensures a reliable and scalable intrusion detection system by including data preparation, model architecture design, training, and evaluation. This stage of the methodology involves data acquisition and preprocessing. The CICIDS 2017 dataset, which includes a wide range of real-world simulated network traffic, was selected due to its comprehensiveness and relevance to current network attack scenarios. The preprocessed and balanced "perfect_Balance_dataset_cic-ids-2017" dataset, which include both benign and malicious traffic, was used in this investigation. Separating the feature set from the labels was the first step in the preprocessing procedure. In particular, the target variable for binary classification—where 0 denotes benign traffic and 1 denotes malicious traffic—was taken from the label_binary column. To make sure that only numerical characteristics were evaluated, non-feature columns like Label and Attack were also eliminated. To ensure that all input features contribute equally to the training of the models and to avoid scale-dominance, feature scaling was performed using the StandardScaler from the sklearn.preprocessing module. This standardization step transforms the data so that each feature has a mean of zero and a standard deviation of one. To ensure appropriate evaluation and prevent data leakage, the dataset was scaled and then divided into training and testing groups using an 80/20 ratio. The core of the feature extraction phase involved designing and training an autoencoder using TensorFlow and Keras. This study's encoder architecture captures the most important aspects of the input by utilizing a bottleneck layer of 32 neurons and two hidden layers with ReLU activation functions. By reversing this structure, the decoder recreates the initial input. Notably, the decoder's last layer employs a linear activation function to guarantee compatibility with standardized input values, which encompass both positive and negative numbers. The autoencoder was compiled using the Adam optimizer and Mean Squared Error (MSE) loss function and trained for 30 epochs with a batch size of 128. This training enables the encoder to learn meaningful compressed representations of the input data. After training, a supervised machine learning model was then fed these encoded features, which reflect the inherent structure of the original high-dimensional data. For this work, XGBoost, a gradient boosting classifier renowned for its effectiveness and strong prediction performance, was used. The encoded training data and the associated binary labels were used to train the classifier. Upon completion of the training phase, the model was tested on the encoded test set. In the last assessment, the XGBoost classifier's performance was gauged using common classification measures. This hybrid model achieved the great results by achieving the accuracy of 98.15%.

4.5.2 Result of Hybrid Model

Dataset	Hybrid Model	Accuracy	Precision	Recall	F1-Score	Confusion Matrix
CICID2017	Deep stacked Autoencoder to learn pattern XGBoost classifier to detect attack	98.15%	0.98	0.98	0.98	[109184 2217] [1880 109342]

4.5.3 Hybrid model representation

The confusion matrix, accuracy score, and classification report (which includes precision, recall, and F1-score) were used to evaluate the predictions made for the test data. These metrics offer a thorough evaluation of the model's accuracy in differentiating between malicious and benign network traffic. This hybrid approach shows how well intrusion detection systems work when deep learning is used for feature extraction and conventional machine learning is used for classification. While XGBoost uses this fine-tuned feature space to achieve great classification performance, autoencoders help with noise reduction and dimensionality compression.


CICIDS2017 Dataset
Preprocessing (Cleaning, Normalization, Feature Selection)
Deep Stack Autoencoder (Training & Feature Extraction)
XGBoost (Prediction)
Final Output (Attack / Normal)

4.6 Development of Restful API

After designing and developing the proposed hybrid model for intrusion detection, the next step was to implement it through a well-structured RESTful API system. This intrusion detection system's technique is set up to offer a thorough, multi-layered strategy for detecting hostile network behavior by combining deep learning, classical machine learning, and interpretability based on huge language models. The system begins with the preprocessing of incoming network traffic data, which is expected in JSON format and is converted into a Pandas Data Frame. Any superfluous columns, such as label or label_binary, are removed to guarantee consistent input for the model. The raw feature values are then scaled using a pre-trained scaler object, typically trained on the same dataset used during model development. By doing this, the input distribution is guaranteed to match the model's expectations. After scaling, the data is passed through a deep learning-based encoder, which transforms the data into a condensed feature representation. Through the capture of nonlinear interactions within the feature space, this encoding stage aids in the generalization of the model. Following transformation, the data is fed into an XGBoost classifier that has already been trained. The classifier produces binary predictions that indicate whether the network flow is categorized as "Attack" or "Benign." In addition, the model generates class probabilities for every input, which are utilized to compute other metrics like classification margin, entropy, and attack confidence scores. Higher entropy levels indicate less confidence predictions, as entropy measures prediction uncertainty. Margin measures the difference between the top two predicted probabilities, giving insight into how distinct the classification decision was. These metrics have two functions: they aid analysts in assessing the level of confidence in each forecast and in deciding how much faith to put in the model's output.

The system also categorizes the severity of each prediction based on the attack confidence score. Scores above 0.9 are labeled as "High" severity, those between 0.7 and 0.9 are marked as "Medium," and anything below is considered "Low." This aids in setting priorities for threat response activities. For every request, the system computes and returns statistics, such as the overall percentage of attacks, the number of records, the number of records classed as assaults, and the number of records classified as benign. It is appropriate for real-time monitoring and analysis dashboards since each prediction result is assembled into a structured response that includes the original input, the prediction label, the confidence score, entropy, margin, and severity level.

An additional and unique layer of the methodology involves integrating a locally running large language model through Ollama, Ollama model give slow and inaccurate response so it is decided to use openai gpt 4 model and build a custom model, trained with prompt engineering, name as ak-cyberguard. With the use of a prompt that includes important parameters such as attack percentage, total flow count, and the number of attack versus benign samples, the system creates a text-based summary or advice based on prediction findings. This prompt is sent to the LLM using a subprocess call, and the model returns a short, professional interpretation or recommendation that can assist security analysts in decision-making. What would otherwise be raw numerical outputs are made more explainable by this layer of natural language interpretability.

The backend is implemented using Django and Django REST Framework. There are several API endpoints available, each with varying levels of information, ranging from straightforward binary forecasts to in-depth model uncertainty analysis and confidence grading. Even in the event that input data is absent or distorted, the endpoints' robustness and error management guarantee dependable functioning. Additionally, there is a chatbot interface that lets users ask natural language inquiries about forecasts. This chatbot may provide simple, contextual responses by using a streamed response from a locally hosted model. Basic frontend rendering features are also provided to support HTML templates, possibly for an online dashboard.

4.7 Deployment of Models on Virtual Machine

After completing the implementation of the API for the hybrid intrusion detection model, the next stage involved a comprehensive sequence of technical preparations and deployments, which was executed to ensure the reliability, scalability, and security of the system. The necessity to integrate the website with pre-trained machine learning and deep learning models that could anticipate cyberthreats was at the heart of this deployment procedure. These models were not trained on the live server environment; rather, they had been trained locally during earlier development phases. However, for these models to be used in real-time by the website's frontend, it was essential to deploy them onto a remote cloud-based environment. This was achieved by setting up a Virtual Machine (VM), which would serve as the secure and dedicated host for model inference. The website could only use backend APIs to receive real-time predictions from the models following this deployment stage. Provisioning a cloud-based virtual machine running Ubuntu 24.04 was the first step in the deployment procedure. This virtual server was assigned a public IP address and was then secured and updated with all essential system-level dependencies required for deploying a Django-based backend application. Before any aspect of the frontend could become functional, the complete Django project, which served as the website's backend, was cloned from its source GitHub repository into this VM environment. A Python virtual environment was created and activated to ensure that the application's libraries and packages were isolated and did not interfere with system-wide installations. Using the requirements.txt file, all required dependencies such as Django, NumPy, TensorFlow, Pandas, and other libraries essential for data handling and model inference were installed. Important adjustments were made to the Django settings in order to get the backend ready for production. Debug mode was turned off by setting DEBUG = False, and both the server's public IP and the registered domain cyberfyp.duckdns.org were added to the ALLOWED_HOSTS list. In order for the Django server to function safely in a production setting and only accept connections from legitimate hosts, these configurations were necessary. Next, the collectstatic command was used to gather all of the Django application's static files, including images, CSS, and JavaScript. This process aggregated all static assets into a centralized location from where they could be efficiently served by the web server (Nginx). Simultaneously, Django database migrations were applied using python manage.py migrate to set up the required database schema and prepare the backend to handle incoming data and model predictions. Before serving any user traffic, the Gunicorn WSGI HTTP server was configured to act as an interface between Django and the Nginx web server. In order to ensure effective communication, Gunicorn was configured to execute through a Unix socket file (fyp.sock). A system service was then created to manage Gunicorn as a background process, enabling it to automatically start on system boot and ensuring high availability of the Django application. With the backend and model infrastructure prepared, the next step was to set up Nginx as a reverse proxy web server. In addition to forwarding all application-related queries, including those aimed at the endpoints /predict/, /pre2/, and /chat/, to the Gunicorn socket, Nginx was set up to serve all static files straight from the gathered directory. These settings were essential to guaranteeing a quick, dependable, and organized data transfer between the virtual machine's backend models and the frontend user interface. DuckDNS, a dynamic DNS provider, was used to register a subdomain so that the application could be accessed easily. The chosen subdomain, cyberfyp.duckdns.org, was configured by updating its DNS A record to point to the VM's public IP address. In order to enable the server to handle traffic sent to cyberfyp.duckdns.org appropriately, the server_name directive in the Nginx configuration was also modified to reflect this domain. At first, browsers showed a security warning stating that the connection was insecure when the website was viewed using conventional HTTP. To resolve this and enable secure communication over HTTPS, a free SSL certificate was obtained from Let’s Encrypt using the Certbot tool. SSL was manually incorporated into the Nginx configuration by directly linking the certificate and key files, notwithstanding the first failure of the automatic configuration. The effective integration guaranteed that all data sent between users and the server was encrypted and secure, and it also removed browser safety warnings. A secure HTTPS connection that complied with current online security standards and increased user trust was the end result. In order to guarantee continuous secure access, Certbot was also programmed to automatically renew the SSL certificate prior to its expiration. With the infrastructure securely in place, including the model deployment, backend APIs, server configurations, domain settings, and SSL encryption, the final step involved connecting the frontend of the website to this backend system. Now, whenever a user enters data into the website interface, whether for file analysis, threat prediction, or live detection, JavaScript records the input, forwards it via AJAX calls to the Django backend, and the backend interacts with the virtual machine’s models. The user is then presented with the prediction results dynamically via the frontend interface after they are returned in JSON format. A comprehensive and technically sound deployme’t infrastructure was set up prior to the website being made available to users. All of the important components, including linking the domain and turning on HTTPS, configuring Django and Gunicorn, setting up Nginx and SSL certificates, and hosting the trained models in a virtual machine, were meticulously set up and tested beforehand. This made sure that in addition to being scalable and secure, the website could handle real-time user requests for AI-powered cyber threat detection.

4.8 Development and Deployment of Dynamic Website

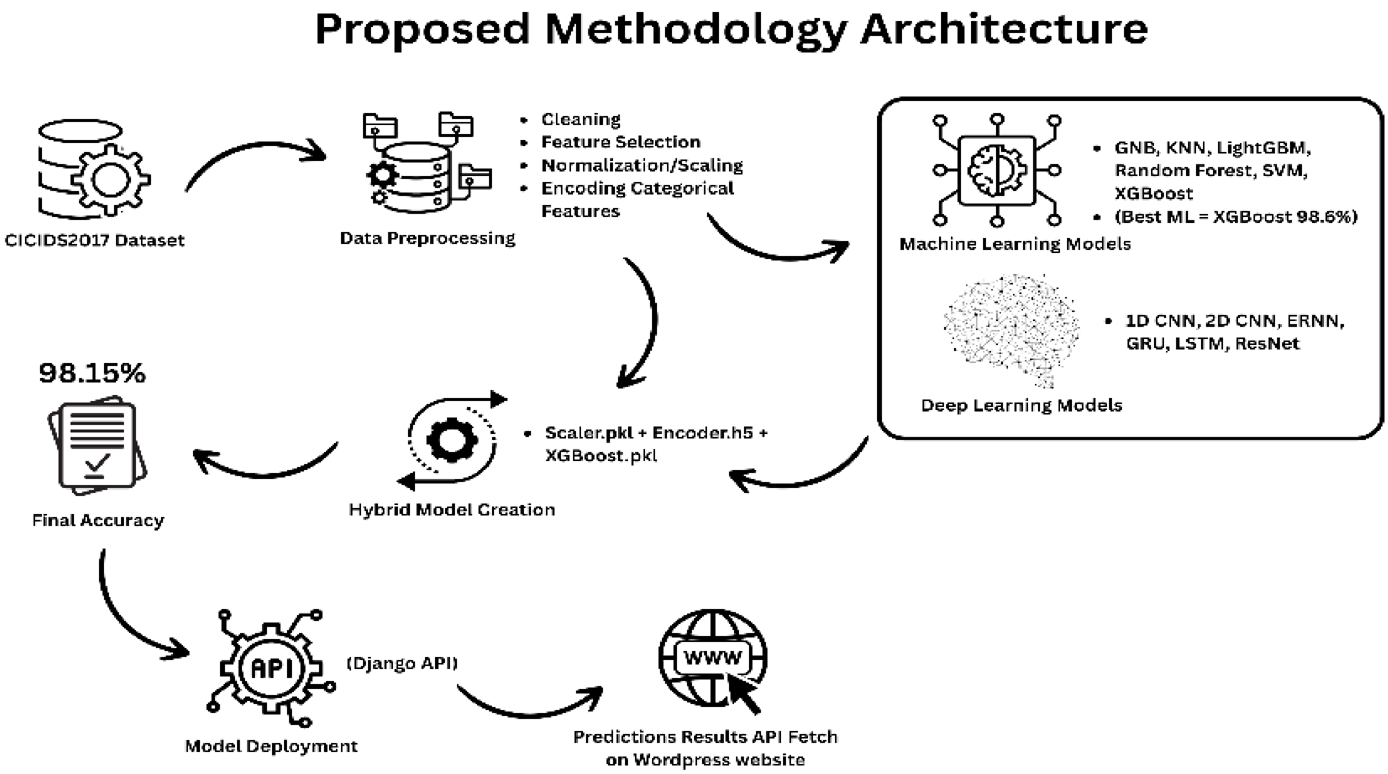
The next stage is to create a fully functional website that combines the trained models via backend APIs when they were deployed on the virtual machine. The website utilizes these APIs to send user input to the deployed models, receive predictions, and generate responses based on the output of the trained models running on the Virtual Machine. This marked the beginning of the website development phase, which required arranging a domain and hosting environment. The hosting service was already available through Hostinger, and the domain in use was “akdeepknowledge.com”. To integrate the final-year project, the name “Cyber Guard” was selected for deployment, and it was configured as a subdomain under the existing domain, resulting in the project being hosted at a subdomain of akdeepknowledge.com. With the domain and hosting ready, the development environment was prepared by installing WordPress on the Hostinger-hosted server. Because of its responsiveness, ease of use, and support for rapid development, WordPress was selected as the content management system. WordPress was installed, and then the required themes and plugins were added to improve the site’s look and feel. HTML, CSS, and JavaScript were employed for specific front-end design and interactivity to create a polished and user-friendly interface, even though WordPress managed the content management and routing of these pages. Five primary pages comprised the website’s structure: Home, APIs, About Us, Services, and Contact Us. The Home page was designed to be the central interactive hub of the website. The LLM Suggestion section, which takes user input, makes predictions using the backend models, and then creates suggestions that are comprehensible by humans using a local language model, is the fourth key component. This section provides a detailed prediction table along with a summary based on the model output.
A description of the project, covering important priority areas like real-time detection, encrypted traffic analysis, and the combination of AI and cybersecurity, is also included on the Home page beneath these features. The architecture of the implemented model is also outlined on this page, followed by statistical insights into system

performance. On the website's second page, "APIs," users can engage directly with the machine learning models by utilizing the prediction APIs that were previously developed using the Django REST Framework. The page provides access to four different prediction endpoints. Depending on their requirements, users can select from any of the numerous prediction types. User input is captured by JavaScript, which then handles the JSON response and sends the request to the relevant Django API endpoint. Once the response is received, JavaScript processes and displays the prediction results in a readable and structured format on the same page. Real-time insights are provided without the need to reload the page thanks to this configuration, which permits dynamic interaction between the frontend and the backend. The third page, "About Us," introduces the developer behind the website and the project. It provides a personal and professional background to help users understand the motivation and expertise driving the development of the Cyber Guard system.

"Services," the fourth page, lists the cybersecurity-related services and features that the system provides, including threat detection, anomaly analysis, and AI-driven advise. Last but not least, the "Contact Us" page offers a way for users to get in touch with us with questions, comments, or requests for help with cooperation or deployment.

Altogether, the methodology for website development followed a structured approach in which a user-friendly frontend was created using WordPress and hosted under the subdomain cyberguard.akdeepknowledge.com. The backend was developed using Django, and the trained machine learning and deep learning models were deployed on a cloud-based Ubuntu virtual machine. These models were made accessible through REST APIs hosted at cyberfyp.duckdns.org. In order to transmit user input data and obtain real-time threat predictions, the frontend and this backend communicate. This interaction is managed by JavaScript, which sends requests to the API, processes the answers, and shows them on the page. The solution offers safe HTTPS communication, realtime monitoring via visualizations, intelligent LLM-based recommendations, and thorough threat detection. Overall, this platform is a comprehensive, deployable, and production-ready solution for handling contemporary cybersecurity concerns since it combines an interactive interface with a scalable backend.

4.9 Architecture of Proposed Methodology in Present Study



Proposed Methodology Architecture
CICIDS2017 Dataset
Data Preprocessing
• Cleaning
• Feature Selection
• Normalization/Scaling
• Encoding Categorical Features
Machine Learning Models
• GNB, KNN, LightGBM, Random Forest, SVM, XGBoost
• (Best ML = XGBoost 98.6%)
Deep Learning Models
• 1D CNN, 2D CNN, ERNN, GRU, LSTM, ResNet
• Scaler.pkl + Encoder.h5 + XGBoost.pkl
Hybrid Model Creation
98.15%
Final Accuracy
API
(Django API)
Model Deployment
WWW
Predictions Results API Fetch on Wordpress website

Chapter 5

Results

The evaluation of the proposed intrusion detection system involved extensive testing using both CICIDS2017 and UNSW-NB15 datasets. The results reflect the performance of both deep learning models and traditional machine learning algorithms, as well as a hybrid system that combines the advantages of both approaches.

On the CICIDS2017 dataset, deep learning models were first applied to assess their ability to detect encrypted threats. The 1D-CNN achieved an accuracy of approximately 93.18%, demonstrating a balanced capability to learn from sequential flow patterns. The 2D-CNN model slightly improved this result with 94.28% accuracy, likely due to its capacity to learn spatial representations. Among recurrent models, the Elman Recurrent Neural Network (ERNN) performed the best, reaching an accuracy of 95.71%, along with stable F1-scores and lower loss values—indicating strong learning of temporal features. Other models like GRU (93.35%), LSTM (93.90%), and ResNet (91%) also showed competent performance but could not outperform ERNN. In terms of traditional machine learning models on CICIDS2017, Naïve Bayes (GNB) failed to perform effectively, yielding only 50.13% accuracy, mainly due to its inability to model complex relationships in flow-based features. On the other hand, K-Nearest Neighbors (KNN) showed strong generalization, while Support Vector Machine (SVM) provided a solid 92.13% accuracy. However, ensemble methods like Random Forest (98.38%), LightGBM (98.60%), and XGBoost (98.62%) outperformed all others. These models delivered exceptional precision and recall, indicating their robustness in handling high-dimensional data and class imbalances. When applied to the UNSW-NB15 dataset, a similar evaluation strategy was followed. After preprocessing and scaling, machine learning models like KNN and SVM achieved 99.22% and 97.02% accuracy, respectively. Among ensemble methods, LightGBM (98.82%), XGBoost (99.04%), and Random Forest (99.22%) once again proved to be the most effective. These results confirm that ensemble methods consistently deliver high performance across multiple datasets. Among deep learning models, LSTM, GRU, and ERNN achieved an average accuracy of 87.37%, showing reliable performance, although still slightly behind the ensemble models. Convolutional models such as 1D-CNN (98.55%) and 2D-CNN (95.65%) also yielded competitive outcomes. Naïve Bayes again underperformed with 81.75% accuracy, further confirming its inadequacy for complex intrusion detection tasks. The most notable performance came from the hybrid model, which combined an autoencoder for feature compression with XGBoost for classification. Trained on a balanced version of the CICIDS2017 dataset, this pipeline achieved exceptional accuracy, precision, and F1-score, leveraging the strength of deep learning to reduce feature dimensionality and the power of XGBoost for final decision-making. The inclusion of LLM-based interpretability also enhanced the practicality of the system, allowing predictions to be summarized and explained in human-readable form. Additionally, a significant result of this study is the successful deployment of the trained models onto a cloud-based virtual machine (VM), along with the development of a fully functional website that interacts with the backend in real-time. The website—hosted on a subdomain and integrated using Django REST APIs—allows users to upload traffic data, receive threat predictions, visualize results via pie charts, and even get LLM-generated summaries. This end-to-end deployment validates the system's usability, scalability, and readiness for real-world applications beyond experimental testing.

5.1 Results of all Machine Learning and Deep Learning Models applied on both

Datasets	Machine Learning Models	Accuracy	Recall	F1-Score	Precision	Confusion Matrix
CICID 2017	Random Forest Classifier	98.38%	0.98	0.98	0.98	[[9709 203] [121 9967]]
CICID 2017	Support Vector Machine (SVM)	92.13%	0.92	0.92	0.92	[[3714 256] [374 3656]]
CICID 2017	K- Neighbors Classifier	98.53%	0.99	0.99	0.99	[[109465 1936] [1339 109883]]
CICID 2017	Guassian Naïve Bayes	50.13%	0.5	0.34	0.7	[[517 166450] [64 166903]]
CICID 2017	Light GBM	98.60%	0.99	0.99	0.99	[[109805 1596] [1512 109710]]
CICID 2017	XGBoost	98.62%	0.99	0.99	0.99	[[109824 1577] [1495 109727]]
UNSW-NB15	Guassian Naïve Bayes	81.69%	0.82	0.81	0.86	[[41653 22768] [768 63325]]
UNSW-NB15	K- Neighbors Classifier	99.04%	0.99	0.99	0.99	[[63471 950] [286 63807]]
UNSW-NB15	Light GBM	99.25%	0.99	0.99	0.99	[[63499 922] [40 64053]]
UNSW-NB15	Random Forest Classifier	99.23%	0.99	0.99	0.99	[[63568 853] [142 63951]]
UNSW-NB15	Support Vector Machine (SVM)	95.45%	0.96	0.95	0.96	[[960 75] [16 949]]
UNSW-NB15	XGBoost	99.25%	0.99	0.99	0.99	[[63525 896] [65 64028]]
Deep Learning Models						
CICID 2017	1D CNN	93.18%	0.93	0.93	0.93	[[103610 7702] [7468 103843]]
CICID 2017	2D CNN	94.28%	0.95	0.95	0.95	
CICID 2017	ERNN	95.71%				
CICID 2017	GRU	93.35%				
CICID 2017	Long Short-Term Memory (LSTM)	93.90%				
CICID 2017	ResNet	90.06%	0.91	0.91	0.91	[[107051 4350] [16367 94855]]

UNSW-NB15	1D CNN	98.55%				
UNSW-NB15	2D CNN	95.64%				
UNSW-NB15	ERNN	87.36%				
UNSW-NB15	GRU	87.36%				
UNSW-NB15	LSTM	87.36%				

Dataset	Hybrid Model	Accuracy	Precision	Recall	F1-Score	Confusion Matrix
CICID2017	Deep stacked Autoencoder to learn pattern XGBoost classifier to detect attack	98.15%	0.98	0.98	0.98	[[109184 2217] [1880 109342]]

In 2024, researchers proposed an intrusion detection system based on a Least Squares Support Vector Machine (LS-SVM) model. Their evaluation on the CICIDS2017 dataset reported an accuracy of 99.5% with precision, recall, and F1-score all close to 100%, showing that the model was capable of highly reliable classification of network intrusions. On the UNSW-NB15 dataset, the same LS-SVM approach achieved 93.3% accuracy with precision reaching 100% and recall at 98%, demonstrating strong but slightly reduced effectiveness compared to CICIDS2017. This study highlights the robustness of LS-SVM for IDS applications and serves as a benchmark for modern machine learning–based detection systems[13].

Another notable 2024 advancement integrated Stacked Sparse Auto-Encoders (SSAE) with a Temporal Convolutional Network (TCN) to analyze time-series network data. On the UNSW-NB15 dataset, this hybrid model achieved an accuracy of 99.43%, with precision at 99.38%, recall at 99.60%, and F1-score at 99.49%, showing that temporal feature learning methods are highly effective at detecting hidden cyber threats within encrypted flows. The model's ability to balance accuracy with strong precision and recall makes it a powerful option for real-time IDS deployment[14].

Also in 2024, an AutoML ensemble study compared several optimized models including Random Forest, Decision Tree, Extra Trees, and XGBoost across standard intrusion detection datasets. On CICIDS2017, the ensemble achieved near-perfect performance, with DT, RF, and ET all recording 99.99% accuracy. On UNSW-NB15, the Extra Trees model stood out with an accuracy of 99.95%, surpassing most traditional methods and demonstrating that automated model selection and ensemble methods can deliver extremely high performance for intrusion detection tasks[1].

From 2023, a robust single-classifier IDS approach was introduced and tested on the UNSW-NB15 dataset, where it achieved an accuracy of 97.77%. While its performance was respectable, it lagged behind ensemble and deep learning methods, showing the limitations of relying solely on a single classifier for such complex network intrusion detection problems[15].

Earlier in 2022, a study focusing on the optimization of IDS performance using deep stacking network models reported much lower results on CICIDS2017. Their reported accuracy was 86.8%, which, while significantly below the performance of modern ensemble and hybrid architectures, illustrates the evolution of intrusion detection research and how more advanced AI methods have greatly improved accuracy over the past few years[16].

Chapter 6

Discussion

The results of this study shows that the hybrid intrusion detection system is very accurate and can be used practically in the encrypted networks. When Hybrid intrusion detection system is tested on CICIDS2017 and UNSW-NB15 datasets so the ensemble based models like XGBoost, LightGBM and Random Forest performed well, indicating good results via accuracy, precision recall and F1 score. These models maintain there stability while handling high dimensional data indicating there generalization ability to be strong. ERNN model is the most accurate among deep learning models. It achieved the highest accuracy and performed well among all deep learning models because it is best in learning the temporal patterns of flow. LSTM and CNN also work well but ensemble methods work a bit better than LSTM and CNN. This shows that traditional classifier with good preprocessing hold an advantage for structured network traffic. The most important result was of Hybrid model, in which autoencoder is used for feature extraction and XGBoost classifier is used for classification. This dataset not just reduce the size and the noise of dataset but also improve the detection performance. This use flow based features just because of this user privacy is safe. Trained model is deployed on Virtual Machine and integrate it with public website. Website predicts the results in real time, website gives the LLM based insights and it gives facility to user to upload their file. This proof that model is not on just testing but in real world applications, it is reliable scalable and practical.

The study of 2021 uses a Deep Stacked Autoencoder and extract the important hidden feature from the CICID 2017 dataset, so the complexity of traffic data reduces. Then Random Forest classifier is used for classification of the features to detect normal and attack traffic. This two stage approach use deep learning to make model understand the features and machine learning for prediction and achieve the accuracy of 87%[17].

In this study a two-stage approach is used in which a deep stacked Auto Encoders are used to understand the patterns of feature present in the CICID 2017 dataset by extracting hidden features but are classified by XGBoost classifier to classify the normal and attack traffic and achieving accuracy of 98.15%.

A recent study in (2025) introduced a combine model in which autoencoder and CNN is used. Autoencoder extract hidden feature after compressing data then CNN process those features to detect attacks. Sampling techniques were used so that the values of dataset handled to detect rare attacks. The binary classification setting in this hybrid pipeline result in accuracy of 99.90% accuracy[18].

In this study, first step was to train traditional models on two datasets, CICID 2017 dataset and UNSW-NB15 dataset. Next to check which model is best performing among both datasets and XGBoost was observed as the best performing model on CICID 2017 dataset. Then the autoencoder is used to learn the feature patterns from the dataset of CICID2107 dataset to extract the important hidden features. Instead of using CNN, this study uses XGBoost classifier for classification because XGBoost is robust and fast for training testing of high dimensional and imbalanced data, XGBoost classifier in result detect the attack and normal traffic from the data and achieving the accuracy of 98.15%.

Chapter 7

Conclusion

The intrusion detection systems catch the attacks from the encrypted network traffic but the accuracy of these systems lack somewhere. Thats why main focus of this study is to create an AI system that can detect attack from encrypted network traffic, accurately. Older studies mostly train and test model on one dataset, due to this a model work well in lab or controlled environments but can't perform well in real world networks. Secondly the ML models in older studies don't have much generalization power that there model perform well on each dataset. This study used a hybrid model which was made by an autoencoder which learn features from the dataset and XGBoost classifier is used for classification which detects the attacks from network traffic. This method only use flow based meta data, for example: size, duration, arrival time and byte rate etc, means the model don't check the content inside a packet, that's why it is compatible for encrypted traffic. To avoid data leakage and overfitting, this study train and test different models on two dataset CICIDS2017 and UNSW-NB15 then every model is checked on different traffic patterns and attack and some performed well. In the last system is deployed on Virtual machine and connected with website where real time prediction is shown and LLM is used which provide insights of the output interpretability and explainable. The results of this study shows that the model's performance is strong. Ensemble models show the best accuracy, more than 98% on CICID2017 dataset and more than 99% on UNSW-NB15 dataset. In deep learning models, ERNN perform well with accuracy of 95.71%. The hybrid model also comes with the strong performance because it performed well classification on compressed dataset. The system is tested on live website after deploying the AI system on Virtual Machines. Users can input there data and can get the real time predictions and LLM will give the human friendly summaries. The results of this research become the base for the future, where detecting encrypted cyberthreats can be done without compromising privacy. Moreover, expanding this concept to make Multiclass classification, real time capture and online learning model which can adapt themselves according to the new patterns. This system can be use in IoT and Software denied Networking SDN environments or can be integrated in the Security Operations Centers SOCs of the big organizations. The modular design of system like preprocessing, model inference, LLM explanation and online interface, gives freedom to researchers to explore the solutions of AI based cybersecurity solutions. This study made a deployable and scalable solution which is practical for encrypted networks and fills the big gap of privacy aware threat detection.

Chapter 8

Reference

[1] Md. A. Talukder et al., “Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction,” Jan. 2024, [Online]. Available: http://arxiv.org/abs/2401.12262

[2] M. Jouhari and M. Guizani, “Lightweight CNN-BiLSTM based Intrusion Detection Systems for Resource-Constrained IoT Devices,” Jun. 2024, doi: 10.1109/IWCMC61514.2024.10592352.

[3] M. A. Alsharaiah et al., “An innovative network intrusion detection system (NIDS): Hierarchical deep learning model based on Unsw-Nb15 dataset,” International Journal of Data and Network Science, vol. 8, no. 2, pp. 709–722, Mar. 2024, doi: 10.5267/j.ijdns.2024.1.007.

[4] M. Farhan et al., “Network-based intrusion detection using deep learning technique,” Sci Rep, vol. 15, no. 1, Dec. 2025, doi: 10.1038/s41598-025-08770-0.

[5] S. Kim, L. Chen, and J. Kim, “Intrusion Prediction using LSTM and GRU with UNSW-NB15,” in 2021 Computing, Communications and IoT Applications, ComComAp 2021, 2021. doi: 10.1109/ComComAp53641.2021.9652926.

[6] Y. Yin et al., “IGRF-RFE: a hybrid feature selection method for MLP-based network intrusion detection on UNSW-NB15 dataset,” J Big Data, vol. 10, no. 1, 2023, doi: 10.1186/s40537-023-00694-8.

[7] G. Apruzzese, L. Pajola, and M. Conti, “The Cross-Evaluation of Machine Learning-Based Network Intrusion Detection Systems,” IEEE Transactions on Network and Service Management, vol. 19, no. 4, 2022, doi: 10.1109/TNSM.2022.3157344.

[8] M. Cantone, C. Marrocco, and A. Bria, “On the Cross-Dataset Generalization of Machine Learning for Network Intrusion Detection,” Feb. 2024, doi: 10.1109/ACCESS.2024.3472907.

[9] G. Sen Poh, D. M. Divakaran, H. W. Lim, J. Ning, and A. Desai, “A Survey of Privacy-Preserving Techniques for Encrypted Traffic Inspection over Network Middleboxes,” Jan. 2021, [Online]. Available: http://arxiv.org/abs/2101.04338

[10] R. A. Elsayed, R. A. Hamada, M. I. Abdalla, and S. A. Elsaid, “Securing IoT and SDN systems using deep-learning based automatic intrusion detection,” Ain Shams Engineering Journal, vol. 14, no. 10, 2023, doi: 10.1016/j.asej.2023.102211.

[11] N. Hassouneh and S. Al-Sharaeh, “Intrusion Detection in IoT Networks using LSTM Deep Learning Models with the UNSW-NB15 Dataset,” in 2025 International Conference on New Trends in Computing Sciences, ICTCS 2025, Institute of Electrical and Electronics Engineers Inc., 2025, pp. 263–269. doi: 10.1109/ICTCS65341.2025.10989358.

[12] F. S. Alsubaei, “Smart deep learning model for enhanced IoT intrusion detection,” Sci Rep, vol. 15, no. 1, Dec. 2025, doi: 10.1038/s41598-025-06363-5.

[13] P. Waghmode, M. Kanumuri, H. El-Ocla, and T. Boyle, “Intrusion detection system based on machine learning using least square support vector machine,” Sci Rep, vol. 15, no. 1, Dec. 2025, doi: 10.1038/s41598-025-95621-7.

[14] Z. He, X. Wang, and C. Li, “A Time Series Intrusion Detection Method Based on SSAE, TCN and Bi-LSTM,” Computers, Materials and Continua, vol. 78, no. 1, pp. 845–871, 2024, doi: 10.32604/cmc.2023.046607.

[15] M. A. Hossain and M. S. Islam, “Ensuring network security with a robust intrusion detection system using ensemble-based machine learning,” Array, vol. 19, Sep. 2023, doi: 10.1016/j.array.2023.100306.

[16] B. Imene, Dr. M. Aissa, and Dr. B. Rafik, “Analyzing and Exploring CIC-IDS 2017 Dataset,” International Journal of Political Science, vol. 9, no. 1, pp. 10–15, 2023, doi: 10.20431/2454-9452.0901002.

[17] N. Fathima, A. Pramod, Y. Srivastava, A. M. Thomas, I. S. P. Syed, and K. R. Chandran, “Two-stage Deep Stacked Autoencoder with Shallow Learning for Network Intrusion Detection System.”

[18] H. Kamal and M. Mashaly, “Enhanced Hybrid Deep Learning Models-Based Anomaly Detection Method for Two-Stage Binary and Multi-Class Classification of Attacks in Intrusion Detection Systems,” Algorithms, vol. 18, no. 2, Feb. 2025, doi: 10.3390/a18020069.

Chapter 9

Appendix (a)

9.1 Pseudo Code of (Hybrid Model with scaler.pkl + encoder.pkl+XGBoost)

```
START
LOAD dataset (CICIDS 2017)
# Step 1: Preprocessing
    - Handle missing values
    - Encode categorical features (Label/One-Hot Encoding)
    - Scale numerical features using scaler.pkl
    - Split dataset into Train and Test sets
# Step 2: Deep Learning Encoder (Feature Extractor)
    DEFINE ERNN/Autoencoder/1D-CNN:
        Input -> Layers -> Latent Representation
    TRAIN encoder on training data
    SAVE encoder as encoder.pkl
    TRANSFORM Train and Test sets using encoder.pkl → Extracted Features
# Step 3: Machine Learning Classifier
    SELECT classifier (XGBoost / Random Forest)
    TRAIN classifier on Encoded Features (Train set)
    SAVE classifier model as model.pkl
# Step 4: Hybrid Inference Pipeline
    INPUT new data sample
    APPLY preprocessing:
        - Load scaler.pkl → scale input
        - Apply encoder.pkl → extract deep features
    LOAD model.pkl → ML classifier
    PREDICT (Benign / Attack)
# Step 5: Evaluation
    CALCULATE Accuracy, Precision, Recall, F1-score
    DISPLAY Confusion Matrix
END
```

9.2 Pseudo Code of (AI-based Anomaly Detection & Threat Analysis Tool)

```
START
// Step 1: Data Preparation
LOAD CICIDS 2017 dataset
CLEAN dataset (remove nulls, duplicates, irrelevant features)
ENCODE categorical features
SCALE numerical features
SPLIT dataset into training set and testing set
// Step 2: Train Models
TRAIN multiple ML models (GNB, KNN, LightGBM, RF, SVM, XGBoost)
TRAIN multiple DL models (1D-CNN, 2D-CNN, ERNN, GRU, LSTM, ResNet)
COMPARE performance of all models
SELECT best models for combination
// Step 3: Build Hybrid Model
SAVE scaler as scaler.pkl
SAVE encoder as encoder_model.h5
SAVE XGBoost as xgboost_model.pkl
COMBINE these models into final hybrid pipeline
CALCULATE accuracy → 98.5%
// Step 4: Deploy Model with Django
CREATE Django API endpoint /predict
WHEN request received:
   - LOAD scaler, encoder, and XGBoost model
   - PREPROCESS input data
   - PREDICT result (Attack or Benign)
   - CALCULATE extra metrics (Entropy, Margin)
   - RETURN JSON response
// Step 5: Frontend Integration (WordPress + JS)
USER uploads dataset (CSV) on WordPress
JAVASCRIPT collects file and sends it to Django API
WAIT for response
DISPLAY prediction results in a table:
   - Each row = record from dataset
   - Show prediction (Attack/Benign)
   - Show entropy and margin
```

```
    - Color code rows for clarity
END
```

9.3 Pseudo Code of (AI Threat Prediction & LLM Suggestion)

```
START
// Step 1–4 (same for all tools)
// Dataset preprocessing → Hybrid model → Deploy Django API
// Step 5: Prediction API
DEFINE Django API /predict_threat
WHEN user input received:
    - PREPROCESS input (scaler + encoder)
    - PREDICT using hybrid model
    - CALCULATE Confidence, Entropy, Margin
    - RETURN response {Prediction, Confidence, Entropy, Margin}
// Step 6: LLM API
DEFINE Django API /llm_suggestion
WHEN activated:
    - RECEIVE prediction results from /predict_threat
    - FORMAT prediction data into prompt
    - SEND prompt to trained LLM model
    - RETURN LLM suggestion {Advice, Mitigation Steps}
// Step 7: Combined API
DEFINE Django API /threat_prediction_llm
WHEN request received:
    - CALL /predict_threat API
    - STORE prediction results
    - IF "Get LLM Suggestion" button clicked:
         CALL /llm_suggestion with prediction results
         RETURN {Prediction Results + LLM Suggestions}
    - ELSE RETURN {Prediction Results only}
// Step 8: Frontend Integration (WordPress + JavaScript)
ON user input (via form upload or manual entry):
    SEND data to /threat_prediction_llm API
    RECEIVE response
    DISPLAY results:
        - TABLE (S.no, Prediction, Confidence, Entropy, Margin)
        - STATS summary (Total Samples, Attack Count, Benign Count, Attack %)
        - BUTTON: "Get LLM Suggestion"
    IF user clicks "Get LLM Suggestion":
        SEND request to LLM API
        RECEIVE AI-generated suggestion
        DISPLAY suggestion text on frontend
END
```

9.4 Pseudo Code of (Live Threat Detection Dashboard)

```
START
// Step 1–4 (same as previous tool “Chat with AI Cyber Expert Advisor”)
// Data Preprocessing → Model Training → Hybrid Model → Django API (/predict)
// Step 5: Create Live Prediction API
DEFINE new API endpoint /predict_live_view in Django
WHEN request received:
    - ACCEPT real-time input (stream or random generated data)
    - PREPROCESS input (scaler + encoder)
    - PREDICT using hybrid model
    - RETURN JSON response: {Time, Prediction, Confidence}
// Step 6: Generate Synthetic Data (for testing)
WRITE Python script to GENERATE random network traffic like ( CICIDS 2017)
SEND this synthetic data to /predict_live_view API continuously
// Step 7: Frontend (WordPress + JavaScript)
INITIALIZE live connection with /predict_live_view API
LOOP:
    SEND request to API
    RECEIVE prediction response
    UPDATE frontend components:
        - PIE chart → attack vs benign ratio
        - BAR chart → attack categories
        - LINE chart → attacks over time
        - TABLE (Time, Prediction, Confidence)
            * Color row based on confidence (light → dark)
        - STATUS BOX (Total, Attacks, Benign, Attack %)
        - NOTIFICATION bell icon if attack detected
DISPLAY updated dashboard in real-time
END
```

9.5 Pseudo Code of (Chat with AI Cyber Expert Advisor)

```
START
FUNCTION ChatWithCyberExpert()
    DISPLAY "Welcome to AI Cyber Expert Advisor"
    LOOP (for each new user query)
        // FRONTEND (WordPress)
        CAPTURE user_input from chatbox
        SEND user_input TO Django_API_ENDPOINT via HTTP POST request
        // BACKEND (Django API)
        FUNCTION Django_API(user_input):
            CLEANED_INPUT = Preprocess(user_input)
            RESPONSE = Call_Model_API(CLEANED_INPUT)
                - Use fine-tuned LLM model
                - Generate cyber security related advice
            RETURN RESPONSE TO WordPress frontend
        END FUNCTION
        // FRONTEND (WordPress)
        RECEIVE RESPONSE from Django API
        DISPLAY RESPONSE to user in chat window
    END LOOP (continues as long as user sends queries)
END FUNCTION
CALL ChatWithCyberExpert()
END
```